Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

INFRA-ESTRUTURA PARA A REENGENHARIA DAS APLICAÇÕES DE UM SISTEMA DE INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA

Marisa Correia de Paiva

Mestrado em Engenharia Informática

2007

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

INFRA-ESTRUTURA PARA A REENGENHARIA DAS APLICAÇÕES DE UM SISTEMA DE INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA

Marisa Correia de Paiva

Projecto orientado pela Prof. Dra. Ana Luísa Respício

e co-orientado por Paulo Azevedo

Mestrado em Engenharia Informática

2007

Declaração

Marisa Correia de Paiva, aluno nº 30047 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado ”Infra-estrutura para a reengenharia das aplicações de um sistema de informação na administração pública”, realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 3 de Setembro de 2007

Paulo Azevedo, supervisor do projecto de *Marisa Correia de Paiva*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado ”Infra-estrutura para a reengenharia das aplicações de um sistema de informação na administração pública”.

Lisboa, 3 de Setembro de 2007

Resumo

Este documento descreve o projecto realizado no âmbito da disciplina Projecto em Engenharia Informática do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa.

A realização deste estágio teve como finalidade um primeiro contacto com a vida profissional na área das novas tecnologias, consistindo na participação num projecto desenvolvido para uma empresa do mesmo sector.

Actualmente, é visível um enorme esforço de simplificação e modernização dos processos na Administração Pública, por parte do governo. Várias medidas foram tomadas, tendo como foco principal os processos organizacionais e os sistemas de informação que os suportam. Foram também propostas um conjunto de inovações que garantem a melhoria da eficácia e da eficiência das organizações.

Associada a esta alteração está, inevitavelmente, um uso mais intenso da tecnologia. Ela serve não apenas para colocar os serviços a comunicar entre si, mas também para simplificar o interface com os utilizadores das aplicações tecnológicas.

Este projecto é desenvolvido numa empresa da Administração Pública, de grande dimensão, através da Novabase. De forma a adaptar-se às medidas e alterações actuais, o projecto consiste em realizar parte da análise e o desenvolvimento de uma infra-estrutura que servirá de base para a reengenharia de um sistema de informação já existente na empresa, na concretização de novas funcionalidades e na adaptação da mesma para novas áreas, nomeadamente o Gabinete Jurídico da instituição.

O desenvolvimento é feito sobre a tecnologia *Windows Forms*, da *Microsoft* e para a sua realização foi utilizada uma *framework* de desenvolvimento da Novabase, a *FrameworkNB*, desenvolvida sobre a *Framework .Net 2.0*, na linguagem C#.

PALAVRAS-CHAVE:

Administração Pública; Sistema de Informação; Reengenharia; Jurídico; *Windows Forms*; *FrameworkNB*

Abstract

This document describes the project carried out in the scope of the discipline “Project in Informatics Engineering” of the Master in Informatics Engineering of the Faculty of Sciences of the University of Lisbon.

The purpose of this period of training was to have a first contact with the professional life in the area of the new technologies, consisting of the participation in a project developed for a company of the same sector.

Currently, it is visible an enormous effort of simplification and modernization of the processes in the Public Administration, by the government. Some actions had been taken focusing the organizational processes and the information systems that support them. It has been also proposed a set of innovations that guarantee the improvement of the effectiveness and the efficiency of the organizations.

Associated to this update it is, inevitably, a more intense use of the technology. It serves not only to place the services to communicate between themselves, but also to simplify the interface with the users of the technological applications.

This project is developed in a great dimension Public Administration company, through Novabase. To adapt it to the actual actions and modifications, the project consists of carrying out part of the analysis and the development of an infrastructure that will be the base for a re-engineering of an information system that already exists in the company, the implementation of new functionalities and the adaptation of the that application for new areas, namely the Legal Cabinet of the institution.

It’s developed in the technology Microsoft Windows Forms and was used a Novabase framework, the FrameworkNB, developed under the Framework. Net 2.0, in the language C#.

KEYWORDS:

Public Administration; Information System; Re-engineering; Legal; *Windows Forms*; *FrameworkNB*

Conteúdo

Lista de Figuras

Capítulo 1

1. Introdução

Este relatório insere-se no âmbito da disciplina de Projecto em Engenharia Informática (PEI) que é parte central do Mestrado em Engenharia Informática (MEI), leccionado pela Faculdade de Ciências da Universidade de Lisboa.

A instituição de acolhimento foi a Novabase, que opera na área das Tecnologias da Informação. Sendo uma empresa de consultoria, o projecto será implementado directamente no cliente que, dados os requisitos de sigilo, será referido como "ABC". O projecto consiste na primeira fase da reengenharia do Sistema de Informação já existente na ABC e no desenvolvimento da aplicação para o Gabinete Jurídico da mesma instituição - JUR.

1.1. Contexto e Motivação do Projecto

O sistema de informação já existe na ABC há 8 anos, ao longo dos quais foi sofrendo várias adaptações. É neste momento constituído por 7 aplicações que servem de suporte ao negócio da empresa. Desde 2005 que têm sido efectuadas diversas intervenções no sentido de optimizar alguns processos de negócio e estender a aplicação existente a alguns sectores de negócio até ao momento sem suporte informático ou com aplicações isoladas.

No entanto, percebeu-se que:

- a agilidade com que a ABC terá que reagir a alterações de negócio será cada vez maior;

- as aplicações existentes dificilmente suportam outras adaptações de grandes dimensões;

- as arquitecturas de Sistemas de Informação e o *software* de base para a implementação dos sistemas tiveram, ao longo destes 8 anos, evoluções consideráveis.

Desta forma, surgiu a oportunidade de fazer uma reengenharia do sistema existente, criando a aplicação SiABC (Sistema de Informação da empresa ABC), de forma a responder com agilidade às necessárias alterações do negócio da ABC e que

simultaneamente melhore a sua capacidade operacional. Essa reengenharia pode ser definida da seguinte forma [1]:

- Exame e alteração de um sistema existente, para o reconstruir numa nova forma e subsequente implementação dessa nova forma;
- Processo de adaptação, de um sistema existente, às mudanças do seu ambiente ou tecnologia, sem alterar o seu funcionamento geral;
- Modificação e futuro desenvolvimento de partes do sistema existente;
- Melhoramento do sistema através do *Reverse Engineering* (e Reestruturação), seguido de *Forward Engineering*, como ilustrado na figura 1.1.

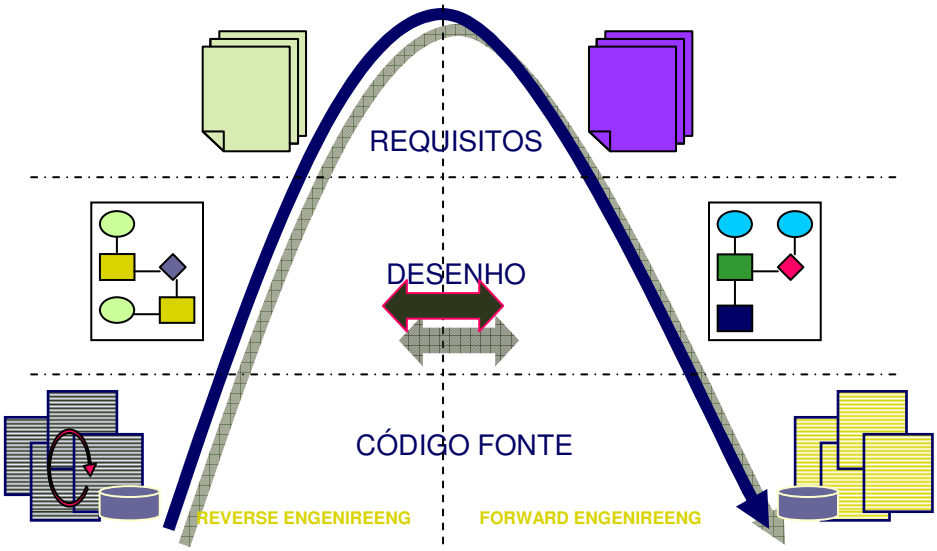

Figura 1.1 - Processo de Reengenharia

Esta reengenharia deverá ser efectuada de forma evolutiva e sem colocar em risco o normal cumprimento das responsabilidades atribuídas à ABC.

1.2. Objectivos do Projecto

Após se perceber a necessidade de tornar as aplicações mais adaptadas ao mundo actual, definiram-se os pontos fracos e as actividades que deverão estar contempladas para os corrigir:

- **Ponto Fraco 1:**

– O actual sistema de informação é constituído por várias aplicações, obrigando nalguns casos à utilização de mais do que uma aplicação por utilizador;

– O acesso a cada aplicação é feito de forma autónoma, o que obriga o utilizador a autenticar-se sempre que muda de aplicação.

Objectivo: Criação de uma única aplicação

– Com a reengenharia do sistema vai ser criada uma única aplicação (SiABC), tendo cada utilizador acesso a todas as funcionalidades e tarefas para as quais está autorizado;

– Revisão da política de acesso à informação (maior granularidade);

– Optimização do número de acessos à base de dados (liberta recursos do Sistema de Gestão de Bases de Dados).

- **Ponto Fraco 2:**

– O actual sistema tem embebido na interface a camada de acesso a dados e um grande número de regras de negócio;

– Existe um grande esforço na manutenção do sistema e, consequentemente, um prazo mais alargado na realização das intervenções necessárias;

– Há um forte constrangimento à reutilização de componentes, obrigando nalguns casos à duplicação de regras, com elevados custos de implementação e sobretudo de manutenção.

Objectivo: Completa reformulação da actual interface

– Separação da camada de interface das regras de negócio e acesso a dados, que irá facilitar o processo de manutenção e promover a reutilização;

– Implementação de um novo interface, com preocupações de melhor usabilidade e aumentos de produtividade.

Objectivo: Optimizações de processos *batch*

Objectivo: Automatização de processos de Negócio.

1.2.1. JUR

Paralelamente à decisão de realizar a reengenharia e de criar o SiABC, o Gabinete Jurídico da empresa manifestou a necessidade de ter também uma aplicação que facilitasse os processos do seu negócio. Havia a hipótese de se criar uma aplicação totalmente nova para este Gabinete, visto que não está directamente relacionado com o negócio da ABC; ou então integrá-lo na reengenharia e na aplicação SiABC, juntamente com todas as outras. Foi escolhida a segunda opção e o projecto JUR – Sistema de Informação para o Gabinete Jurídico – servirá, assim, de piloto, tanto ao nível da metodologia como ao nível da interface, para a reengenharia das aplicações já existentes na ABC.

Após um levantamento da situação actual e uma primeira análise de requisitos, realizada antes do início da minha participação no projecto, chegou-se à conclusão que o JUR tinha como principais objectivos:

- Suportar os processos de negócio do Gabinete Jurídico;
- Manter o registo completo da informação do processo;
- Associar documentos recebidos e/ou produzidos a um processo já existente;
- Possibilitar a utilização de ferramentas específicas para emissão de Estatísticas e demais indicadores de gestão de forma fácil e dinâmica;
- Controlar o ciclo de vida dos processos de negócio:
  - Garantir que o processo segue um circuito pré definido;
  - Tramitar entre actividades de forma automática e controlada;
- Controlar os prazos para a elaboração de tarefas:
  - Enviar automaticamente *e-mails* de lembranças, de acordo com certos critérios;
  - Parametrizar acções automáticas em caso de expiração de prazo;

1.3. Planeamento Inicial

Inicialmente, o trabalho que eu iria desenvolver na empresa estava directamente relacionado com o JUR e, se tudo corresse como planeado (Anexo 1 – Planeamento Inicial), a elaboração do relatório iria coincidir com a entrada em produção da aplicação JUR. Eu teria assim oportunidade de passar por todo o processo de desenvolvimento de uma aplicação, o que seria uma mais valia para apresentar neste relatório.

Como as aplicações já existentes na empresa ABC são o suporte para grande parte do negócio da mesma têm, na maioria das vezes, a prioridade máxima sobre a resolução de problemas que possam surgir. Desta forma, as entregas planeadas para o JUR e para o processo de reengenharia foram frequentemente adiadas por parte do próprio cliente, pois situações mais críticas sobrepuseram-se. Perante esta situação, as minhas tarefas incidiram substancialmente na construção da infra-estrutura que servirá para toda a reengenharia e, consequentemente, para o JUR.

1.4. Organização do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Integração na Instituição Externa:

Capítulo onde é descrito o meu percurso na empresa à qual fui alocada. É descrita a sua história e as principais áreas em que se divide e a área onde este projecto se insere. É feita uma descrição geral da empresa cliente onde o projecto foi desenvolvido e os papéis, responsabilidades e objectivos definidos para a duração do projecto;

- Capítulo 3 – Metodologia e Tecnologias:

Capítulo do documento onde é feita a descrição pormenorizada do ciclo de vida de desenvolvimento dos projectos na Novabase e, mais concretamente, deste trabalho. Contém a calendarização do projecto e são também descritas as tecnologias utilizadas para as tarefas efectuadas;

- Capítulo 4 – Enquadramento:

Capítulo onde se descreve as diversas componentes da arquitectura das aplicações, a *framework* da Novabase utilizada e sua arquitectura e as regras de construção de ecrãs que foram definidas para este projecto;

- Capítulo 5 – Análise e Desenvolvimento:

Este é o capítulo específico do projecto onde estão descritas as tarefas por mim realizadas ao nível da análise, do desenho e da construção e problemas que encontrei;

- Capítulo 6 – Conclusões:

Capítulo onde faço um balanço geral do projecto, especificando aspectos positivos e negativos.

Capítulo 2

2. Integração na Instituição Externa

O meu percurso na Novabase iniciou em Julho de 2006 e durante 2 semanas frequentei uma formação de *Basic Consulting Skills*, em que abordámos temas como:

- Trabalho em equipa;

- Técnicas de apresentação;

- Lidar com o cliente;

- Preparação de reuniões.

Em Setembro recebi formação na empresa Rumos, com componentes mais técnicas, mas ainda uma abordagem geral, sobre:

- Ciclo de vida do *Software* na Novabase;

- 2 *Frameworks* de desenvolvimento utilizadas pela Novabase (com uma das quais irei trabalhar);

- Conceitos gerais sobre Arquitecturas de *Software*;

- *Business Process Management* e *Enterprise Aplication Integration*, duas áreas da Novabase.

Após estas semanas de formação, iniciei o projecto directamente no cliente.

2.1. A Novabase

A Novabase é uma empresa portuguesa de Tecnologias de Informação, criada em 1989. Está dividida nas grandes áreas seguintes: ***Consulting***, ***Engineering***, ***Digital TV*** e ***Capital***, sendo a ***Consulting*** aquela onde fui inserida.

A ***Consulting*** tem o foco na aplicação prática da tecnologia para a resolução dos desafios do cliente. Esta área concebe e implementa soluções de Tecnologias de Informação e possui um conjunto de práticas nos domínios de [2]:

- *Financial Services*;

- *Government & Healthcare*;

- *Telecommunications & Media*;

- *Business & IT Consulting*;

- *Advanced Custom Development*;

- *Business Intelligence*;

- *Enterprise Applications*;

- *Integration e Multioutsourcing Services*.

A área de *Advanced Costum Development* é onde se incluem os projectos que necessitem de construção de *software* à medida de cada cliente, tendo como suporte as *Frameworks* de desenvolvimento Novabase.

2.2. O Cliente

A instituição "ABC", para a qual será desenvolvido o projecto ao qual se refere este relatório, é uma empresa da área da Adiministração Pública. A Novabase opera directamente no Gabinete de Organização e Informática do seu Departamento de Apoio.

2.2.1. Papéis, responsabilidades e objectivos

Como parte da política da Novabase, anualmente são definidos os objectivos e responsabilidades de colaborador, que servem de base para avaliações de desempenho. Como tal, no início da minha colaboração na empresa cliente foram estipulados alguns papéis/responsabilidades/objectivos que eu deveria seguir de modo a cumprir os objectivos definidos para o meu nível de carreira. Os específicos para o projecto em questão, são:

- Domínio da tecnologia e metodologia utilizada na Reengenharia;

- Conseguir a normalização/estandardização dos componentes (ecrãs) realizados na reengenharia;

- Passar/adquirir conhecimento do negócio associado às tarefas realizadas.

Capítulo 3

3. Metodologia e Tecnologias

3.1. Metodologia Novabase

Sendo uma empresa líder de mercado em Tecnologias de Informação e sendo certificada em termos de qualidade, a Novabase tem o dever e a preocupação de implementar as melhores práticas no domínio da engenharia de *software*.

Desta forma, toda a área de *Advanced Costum Development* conhece e implementa uma metodologia comum de desenvolvimento de projectos.

O *Software Development Life Cycle* é definido como "o enquadramento alto nível do processo de desenvolvimento de soluções, das diferentes actividades, dos diferentes papéis e responsabilidades" [3]. Assim, essas actividades serão sempre aplicadas em todos os projectos.

Por ser uma boa prática de referência, a Novabase implementa o Modelo Evolucionário de Processo, que reúne a natureza iterativa da prototipagem com os aspectos sistemáticos do modelo sequencial linear [3], como se pode verificar na figura 3.1:

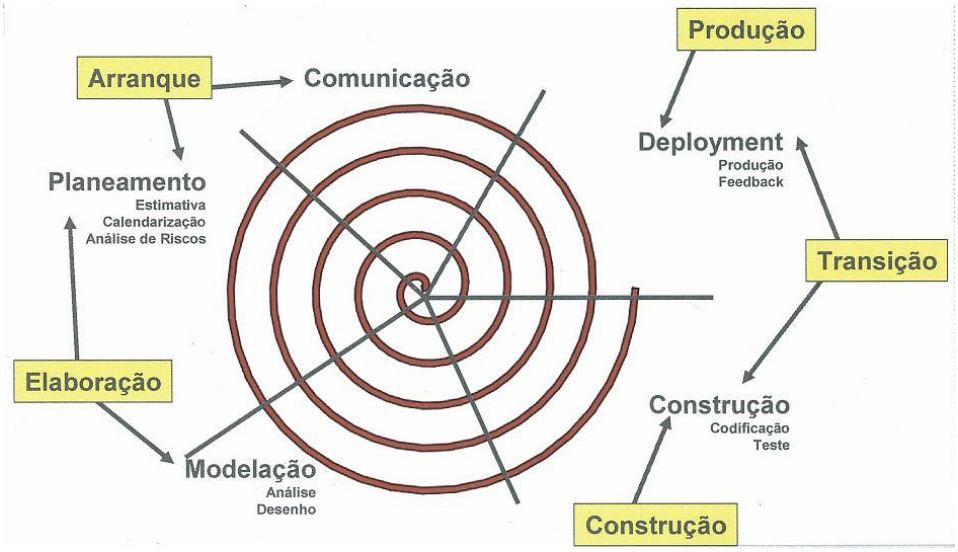

Figura 3.1 - Modelo em Espiral

Apoiando-se neste modelo, a construção de *software* na área de *Advanced Costum Development* funciona de acordo com a figura 3.2. As minhas funções incidiram sobre o que está assinalado nessa figura.

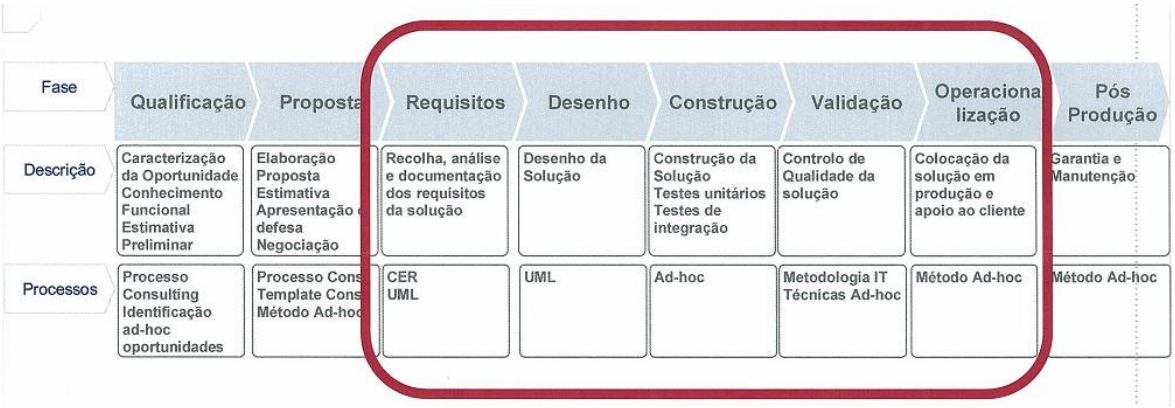

Figura 3.2 - "Máquina" produtiva da área de *Advanced Costum Development*

3.2. Objectivo da Metodologia

Para o caso específico da ABC, a metodologia a utilizar deverá ser baseada no paradigma "Orientado a Objectos" e deverá suportar o ciclo de vida do Sistema de Informação da ABC. É na metodologia que são definidos aspectos como:

- Artefactos do modelo;

- Ferramentas a utilizar;

- *Deliverables* a produzir.

Para corresponder aos objectivos do cliente em relação à aplicação a ser desenvolvida, já referidos atrás, todas as fases da metodologia têm de ter presente certos princípios e aspectos que deverão ser observados na versão final da aplicação, tais como:

- A solução deve ser simples de alterar;

- Deve ser possível introduzir novas funcionalidades (serviços) de modo simples;

- O modelo de dados é único para a organização;

- A plataforma deve permitir a coexistência da solução em produção com a nova solução;

- A metodologia é única e é integrada para especificação e implementação – utilização de UML;

- Deverão haver novos alinhamentos estratégicos:

- Orientação ao processo;

  - Utilização do *Workflow* + Serviços;

- Orientação ao serviço;

  - Utilização da *Service Oriented Architecture* (ver Secção 4.1);

- Orientação ao objecto:

  - Utilização da *FrameworkNB* (ver Secção 4.2);

Ciclo de Vida do *Software*

Um *modelo de processo* é um esquema que organiza, ordena e relaciona a forma como as várias fases e tarefas devem ser prosseguidas ao longo do ciclo de vida do sistema. A função principal de um modelo de processo é determinar a ordem das fases envolvidas no desenvolvimento de sistemas e estabelecer os critérios de transição para progredir entre fases. Na figura 3.3 podemos observar o modelo de processo em cascata, que representa todas as fases de desenvolvimento deste projecto e que serão descritas seguidamente.

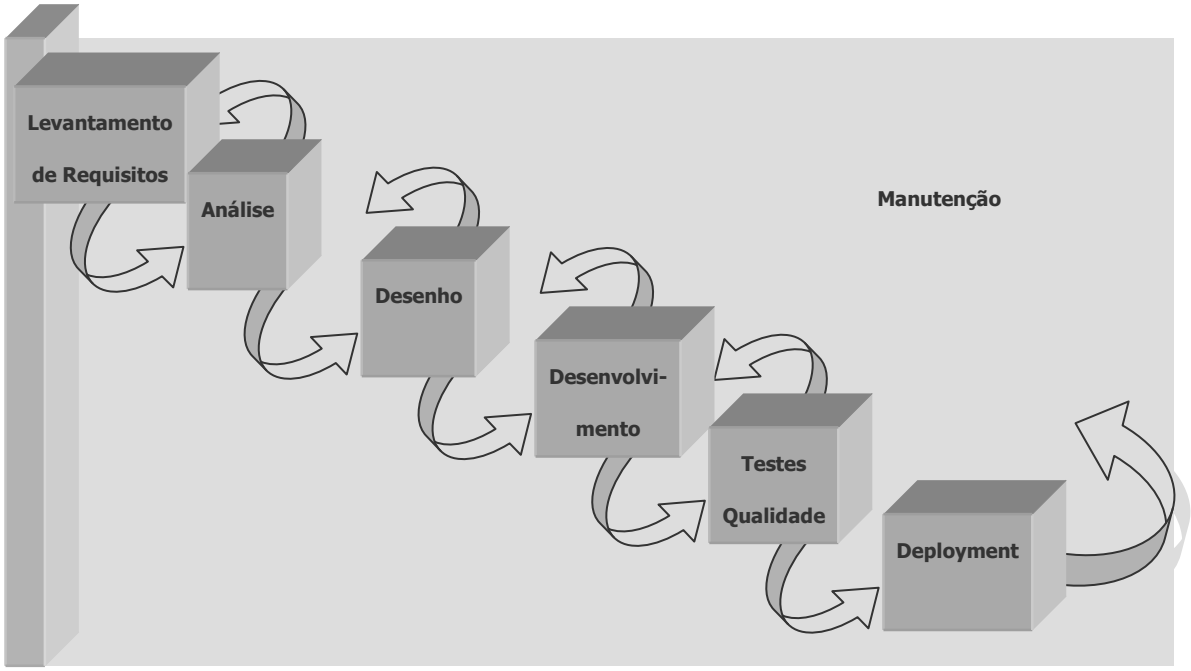

Figura 3.3 - Modelo em Cascata

3.3. Levantamento de Requisitos

A comunicação e iteração com o cliente e a recolha e compreensão dos requisitos de negócio são as actividades *pivot* em todo o ciclo de vida de desenvolvimento de *software* [3]. É sabido que a melhor tecnologia, a melhor aplicação ou a melhor solução podem falhar se não compreendermos o problema do cliente. Assim, esta fase tem como objectivo perceber e documentar "o que o cliente precisa".

Um **requisito** é uma condição ou característica necessária para um sistema alcançar um determinado objectivo ou finalidade.

Um **requisito funcional** especifica acções que o sistema deve executar independentemente de exigências físicas ou tecnológicas. Os requisitos funcionais estão associados ao modelo de negócio, ou seja, são o conjunto das necessidades do cliente que devem ser satisfeitas para resolver um problema ou alcançar um objectivo no seu negócio.

Um **requisito não funcional** especifica as características do sistema ou do ambiente em que ele está inserido. Os requisitos não funcionais referem-se a factores humanos como estética, interfaces amigáveis, documentação e ajuda na Internet ou a factores de desempenho ou segurança, entre outros.

No final desta fase, os *deliverables* produzidos correspondem a:

- Um Modelo de requisitos, onde estão representados os requisitos e as relações entre eles;

- Um documento *Word*, denominado Caderno de Especificação de Requisitos [4].

3.4. Análise (Modelo de Casos de Uso)

Um **caso de uso** representa conceptualmente o conjunto de funcionalidades que o sistema deve possuir para atender aos requisitos do cliente. Serve de "contrato" entre o cliente e o fornecedor. Todos os requisitos funcionais existentes devem ser satisfeitos por um ou mais casos de uso. Um caso de uso pode, em simultâneo, satisfazer mais do que um requisito funcional e podem ser restringidos por um ou mais requisitos não funcionais.

Após termos definido os requisitos isoladamente, é nesta fase que dizemos quem é o actor.

Um **actor** é um utilizador do sistema, ou seja, é um utilizador das funcionalidades disponibilizadas pelo sistema. Ele interage com os casos de uso e pode assumir vários papéis, tais como Humano, máquina, peça de outro sistema ou peça do próprio sistema.

O comportamento dinâmico de um caso de uso é apresentado através de um diagrama de actividades.

Num diagrama de actividades pode modelar-se o conjunto de operações (**Actividades**) a serem realizadas para executar um caso de uso. São apresentados os diversos caminhos lógicos para a execução das actividades representando os paralelismos de acção, pontos de decisão e reacções a eventos [4].

Nesta fase, os casos de uso estão isolados, ainda não se define o caminho a seguir ("vai daqui para ali") mas sim o "faço primeiro isto e depois aquilo".

Como *deliverables*, esta fase produz:

- Um Modelo de Casos de Uso, que contém os casos de uso e actores, as relações entre estes e os Diagramas de Casos de Uso e de Actividades;

- O Caderno de Especificação de Requisitos revisto;

- Um documento Word, denominado Caderno de Análise Detalhada [4].

3.5. Desenho

O objectivo desta fase é a identificação e caracterização das entidades envolvidas no sistema e as suas relações. Assim, é nesta fase que se especificam as tabelas necessárias para suportar a persistência das classes identificadas (modelo lógico e físico) e que se definem as actividades a realizar pelo utilizador, o desenho dos ecrãs, dos relatórios e dos procedimentos necessários.

3.5.1. Modelo de Classes

Uma **classe** descreve um conjunto de objectos que compartilham as mesmas especificações de atributos, operações, restrições e semântica. A finalidade de uma classe é caracterizar a estrutura e comportamento de um objecto.

Os objectos de um sistema são extraídos a partir dos casos de uso e respectivas actividades, identificando as entidades envolvidas. Assim, e como a metodologia é baseada no paradigma "Object-Oriented", cada classe é única no sistema e reutilizável em qualquer contexto [4].

Os **diagramas de classe** apresentam diversas visões do sistema, representando as classes e as respectivas relações de acordo com a referida visão.

Os **diagramas de sequência** (ou interacção) são diagramas dinâmicos que representam o ciclo de vida das instâncias das classes de determinado diagrama de classes [4]. Nos diagramas de sequência é demonstrada a utilização das operações definidas, ao longo do tempo, no âmbito de cenários representativos da utilização do sistema.

3.5.2. Modelo de Dados

De forma a suportar as alterações do modelo de dados sem implicar alterações no modelo físico existente, uma das fases do desenho consiste em criar um **Modelo Lógico.** Este caracteriza-se pelo conjunto de tabelas e relações (esquema de tabelas) derivado directamente do modelo de classes. O conjunto final das tabelas (em nº) pode não coincidir com o conjunto de classes existentes por questões de normalização (3ª Forma Normal) do modelo de dados [4]. Este modelo serve de suporte temporário (em memória) dos dados manipulados na interface e é utilizado pela *Framework .Net* em forma de *DataSet's* (Subsecção 4.2.2)

O **Modelo Físico** é o conjunto de tabelas e relações (esquema de tabelas) derivado directamente do modelo lógico, no caso de novas classes, ou derivado do esquema físico actual no caso de classes existentes. Serve de suporte definitivo (SGBD-Oracle) dos dados do sistema.

Este modelo físico deve evoluir, através da reengenharia, no sentido de suportar o modelo lógico sem mapeamentos. As operações de transferência de dados entre o modelo físico e o modelo lógico (*insert*, *select*, *update*, *delete*) são acrescentadas ao conjunto de operações de cada classe. Estas operações suportam o mapeamento entre os modelos enquanto houver distinções [4].

3.5.3. Interface Model

Nesta fase é também necessário definir as **interfaces** que permitirão a interacção dos utilizadores com o sistema. O modelo de interfaces permite especificar o conjunto de interfaces adequadas para a realização de cada **actividade**.

As actividades do sistema são identificadas com base em:

- Acções que compõem os processos;
- Manutenção dos conceitos ou outras operações sobre os mesmos não enquadradas num processo;

As actividades são caracterizadas por:

- Objectivo e âmbito;
- **Ecrãs** a utilizar;
- Fluxo de navegação entre os ecrãs utilizados;
- Regras de utilização (para posterior definição dos perfis de acesso);
- **Relatórios** a produzir;
- **Procedimentos** a executar.

As actividades são únicas e reutilizáveis. Em seguida, descrevem-se as várias interfaces que realizam as actividades.

Ecrãs

Os ecrãs do utilizador derivam das actividades que é necessário realizar. Um ecrã deve ser direccionado ao conceito (ou detalhe do mesmo), utilizado isoladamente na actividade (manutenção) ou utilizado no âmbito de uma acção.

Para cada ecrã especifica-se:

- Caracterização;
- *Layout*: definição dos componentes a incluir e a sua localização e aspecto;
- Objectos de dados (campos): relacionamento com o modelo de dados (modelo lógico) – *DataSet*;
- Botões e Menus: relacionamento com o modelo de classes (operações) – *Business Objects*;

Os ecrãs podem ser reutilizados por diferentes processos.

Relatórios

Os relatórios derivam igualmente das actividades que é necessário realizar. Podem ser direccionados ao conceito (ex.: Lista de Nacionalidades ou Informação sobre as Entidades), ou direccionados ao processo (ex.: oficio X, certidão Z). Os relatórios deverão ser genéricos, usando parâmetros de entrada para a especialização.

Para cada relatório especifica-se:

- Caracterização;
- Layout: definição da forma do relatório;
- Tipo: definição da tecnologia a usar na criação do relatório (Crystal Reports, VIPP, word);
- Conteúdo: definição da forma de obtenção dos dados variáveis, caso necessário, do relatório. (Associação ao modelo de dados);

Os relatórios podem ser reutilizados por diferentes processos.

Procedimentos

Os procedimentos derivam das actividades que são realizadas sem a intervenção do utilizador durante a sua execução. São vulgarmente conhecidos por "processos *batch*" e as actividades automáticas do *workflow*.

Para cada procedimento especifica-se:

- Caracterização;
- Comportamento do procedimento ou identificação de procedimentos existentes que serão invocados;
- Tecnologia a utilizar para a implementação do procedimento.

Como *deliverables*, a fase de Desenho produz:

- Um Modelo de Classes, um Modelo de Dados e um Modelo de Interfaces;
- O Caderno de Especificação de Requisitos e o Caderno de Análise Detalhada Revistos;
- Caderno de Especificação Técnica;
- Um protótipo da aplicação, a ser apresentado e aprovado, que contenha o *layout* e a navegação entre todos os ecrãs que façam parte do conceito ou do processo, ainda sem ligação à base de dados e ao *workflow* [4].

3.6. Desenvolvimento

O desenvolvimento consiste na concretização propriamente dita. É feita de acordo com os *standards* de desenvolvimento da Novabase e no caso concreto deste projecto, os aspectos abordados são apresentados no Capítulo 4 – Trabalho Realizado.

3.7. Tecnologias Utilizadas

A implementação do projecto foi realizada em Microsoft Visual Studio .NET 2005 na linguagem C#.

Análise e Desenho

Para documentar todo o ciclo de vida do *Software*, era necessário optar por uma ferramenta CASE (*Computer-Aided Software Engineering*):

Ferramentas Case

Uma ferramenta CASE é um produto baseado em computadores com o objectivo de suportar uma ou mais actividades da Engenharia de *Software* num processo de desenvolvimento de *software* [5]. Estas ferramentas auxiliam as actividades de criação e gestão dos produtos elaborados no processo de desenvolvimento de *software*. Têm como objectivo automatizar as tarefas anteriores à construção, tais como Casos de Uso, Diagramas de Classes e Diagramas Entidade-Relação. Assim, o analista utiliza estas ferramentas para elaborar tais diagramas, de forma a facilitar a documentação e o desenvolvimento dos projectos.

Um dos deveres de uma ferramenta CASE é englobar todas as etapas do Ciclo de Vida do *Software*. Por ser uma ferramenta CASE comercial com qualidade que rivaliza (e supera frequentemente) "players" tradicionais tais como IBM *Rational Rose* ou *Borland Together*, porém a uma fracção do custo destas ferramentas [6], apostou-se no *Enterprise Architect* da *Sparxs*, para auxiliar em toda a Metodologia.

Enterprise Architect:

O *Enterprise Architect* combina o poder da especificação do último UML 2.1 com uma performance alta, interface intuitiva, de forma a permitir uma modelação avançada para os projectos e para toda a equipa de desenvolvimento. O *Enterprise Architect* é uma

ferramenta progressiva que suporta todos os aspectos do ciclo de vida do desenvolvimento. Como veremos mas à frente, esta ferramenta, tem uma funcionalidade que permite manter a relação entre todas as fases do ciclo, isto é, é possível a qualquer momento verificar se todos os requisitos estão definidos nos casos de uso, se todos os casos de uso têm classes que os implementam, etc. Para além disso, também suporta testes, manutenção e controlo de alterações [7].

Para a especificação da construção e da documentação de artefactos de sistemas, utilizou-se a linguagem UML (*Unified Modeling Language*) [8]. O UML é uma notação standard de diagramas para desenhar ou apresentar figuras (com algum texto) relacionadas com *software* – principalmente *software* Orientado a Objectos [9].

Num nível mais baixo e a suportar a notação UML está o UML *meta-model* que descreve a semântica dos elementos modelados.

Aplicações Multiple-Document Interface (MDI)

Apesar de mais tarde se ter verificado que não iria ser necessário, no início do projecto optou-se por utilizar as Aplicações Multiple-Document Interface, por permitirem abrir mais do que uma janela dentro de outra janela.

As aplicações MDI permitem mostrar vários documentos simultaneamente, com cada documento a aparecer na sua própria janela [10].

Têm uma única janela principal (a janela pai) que contém uma série de janelas na sua área de cliente (janelas filhos). Cada janela filho é um ecrã que está limitado a aparecer só com o pai. Os filhos normalmente partilham a barra de menu, a barra de ferramentas e outras partes da interface do pai. As janelas secundárias como por exemplo as caixas de diálogo não são limitadas pela área de cliente da janela pai.

Menus MDI Standard

Tipicamente, uma aplicação MDI tem um menu Windows que permite ao utilizador organizar as janelas abertas em mosaico ou em cascata. Este menu também permite a navegação para qualquer das janelas abertas. Um exemplo deste tipo de aplicações é o *Microsoft Excel*, como se pode verificar na figura 3.4.

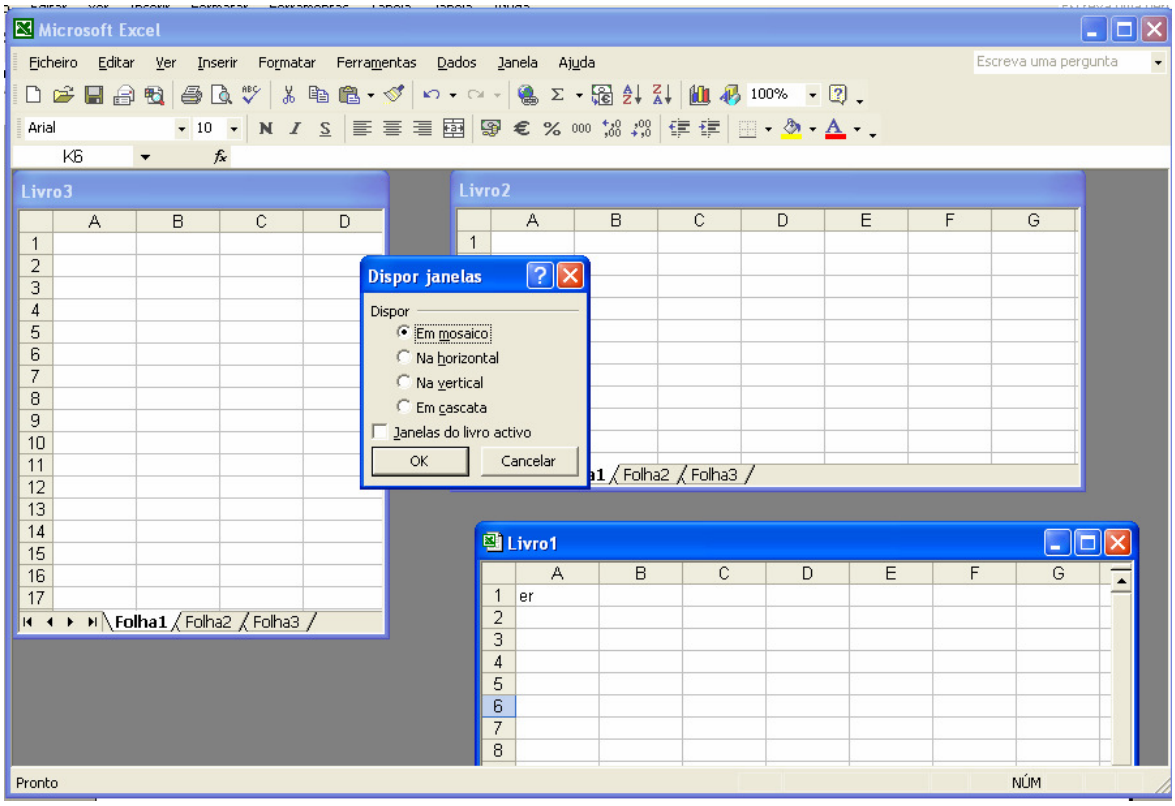

Figura 3.4 - Exemplo de uma aplicação MDI

Base de Dados

Toda a estrutura da ABC e todas as aplicações desenvolvidas na mesma, utilizam como repositório de dados o **Oracle – SQL**. O SQL (*Structured Query Language*) é uma linguagem que fornece uma interface para sistemas de bases de dados relacionais. Esta foi desenvolvida pela IBM na década de 70 e é considerada norma-padrão, tanto como ISO e ANSI. Esta linguagem inclui DDL (*Data Definition Language*), usado para criar e modificar tabelas e outras estruturas da base dados, como por exemplo *views* e sequências, DML (*Data Manipulation Language*), usado para efectuar a gestão dos dados e DCL (*Data Control Language*) como a vertente de controlo de dados.

Para manipular esses dados, utilizou-se o **PL/SQL** (Procedural Language/ Structured Query Language), que é uma extensão procedimental à linguagem de base de dados SQL da Oracle. Esta linguagem suporta variáveis, condições, arrays, cursores e excepções. Permite a criação de uma interface com a base de dados relacional de uma forma imperativa, podendo qualquer query SQL chamar um procedimento/função PL/SQL, ou então que um evento DML dispare um trigger PL/SQL. Os processos PL/SQL, que podem

ser funções, procedimentos, pacotes ou triggers, executam eventos DML e são compilados nas bases de dados Oracle.

Para o desenvolvimento de SQL e PL/SQL em Oracle utilizou-se a ferramenta **TOAD** (versão 8.6). Esta fornece um ambiente cliente gráfico baseado em Windows para o desenvolvimento e administração de recursos em Base de Dados Oracle. O TOAD possui um editor SQL, um editor de Procedimentos (PL/SQL) com um formatador integrado, um navegador de Bases de Dados com todo o tipo de informações e diversas ferramentas associadas a cada tipo de recurso ou objecto.

Controlo de Versões

Para o controlo de versões utilizou-se o ***Visual Source Safe***, que é o *software* de controlo de versões da Microsoft. Esta ferramenta pode ser usada com qualquer tipo de ficheiro produzido por qualquer linguagem de programação, ferramenta de desenvolvimento ou aplicação. Permite aos utilizadores trabalhar ao nível dos ficheiros e dos projectos enquanto promove a reutilização de ficheiros. As suas funcionalidades são orientadas ao projecto e dessa forma aumentam a eficiência da gestão das tarefas do dia-a-dia associadas com *software* baseado em equipa [11].

3.8. Calendário do Projecto

Outubro a Novembro de 2006

- Formação;
- Implementação da Infra-Estrutura base;

Dezembro 2006 a Junho 2007:

- Construção dos Ecrãs Base (Dezembro 2006);
- Construção da Lista de Tarefas (Janeiro 2007);
- Construção do Módulo de Consultas (Janeiro 2007);
- Elaboração do Documento de Apresentação da Aplicação (Fevereiro 2007);
- Protótipo das Juntas Médicas (Fevereiro 2007)
- Protótipo do JUR (Fevereiro / Março 2007)

- Formação *Enterprise Architect* (Março 2007)
- Análise, Desenho e Construção dos Conceitos Associados ao JUR:

  - Parametrização (Abril 2007)
- Análise e Desenho do Módulo de Consultas (Maio 2007);
- Análise e Desenho da Lista de Tarefas (Maio 2007);
- Construção do Protótipo do processo "Elaborar Parecer" (Junho 2007)
- Disponibilização do Protótipo "Elaborar Parecer" (Junho / Julho 2007)

Agosto / Setembro 2007:

- Elaboração do Relatório

Na figura 3.5 é possível visualizar o mapa de *Gant* com as minhas tarefas no projecto.

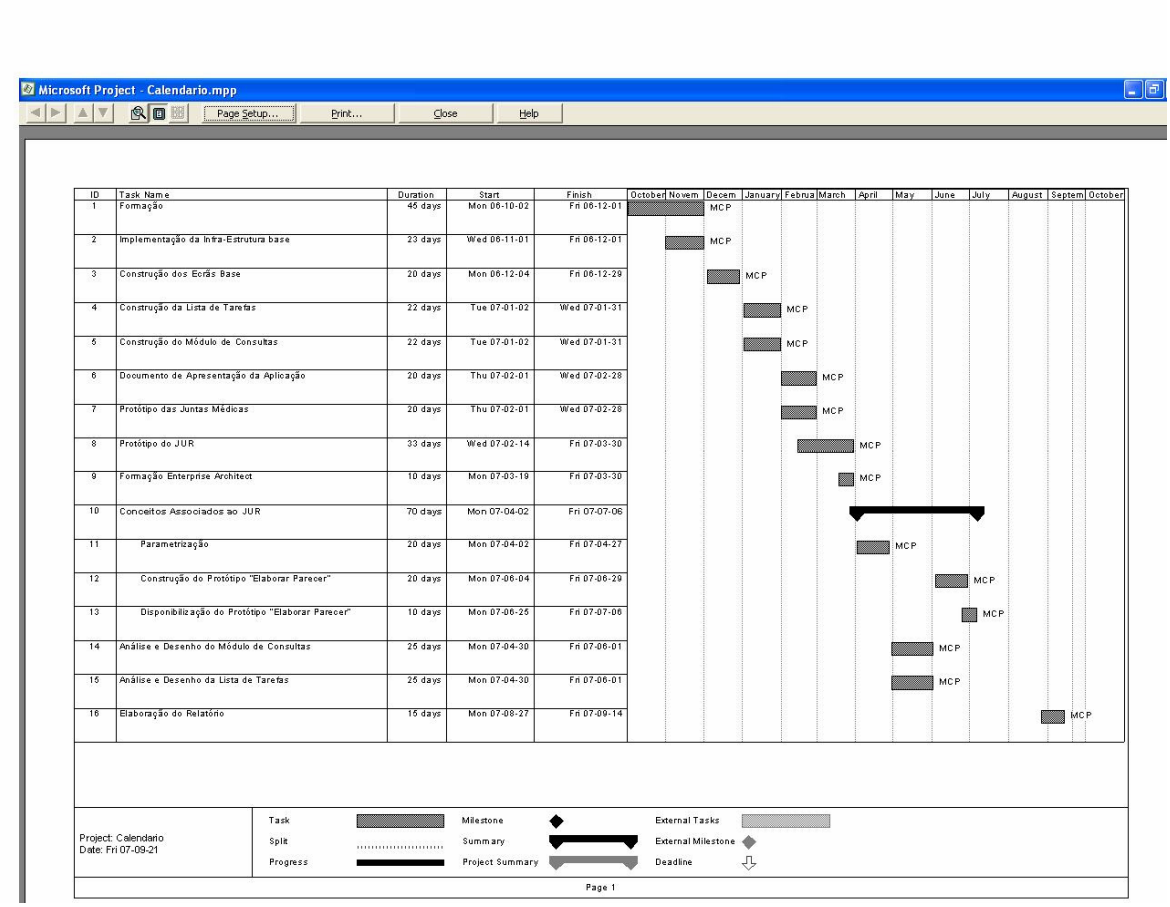

Figura 3.5 - Mapa de *Gant* das minhas tarefas no projecto.

Capítulo 4

4. Enquadramento

4.1. A Arquitectura das aplicações na ABC

De acordo com a *SOA-Reference Model Specification* [12], SOA (*Service Oriented Architecture*) é um paradigma para organizar e utilizar capacidades distribuídas que podem estar sob controlo de diferentes entidades. Esta definição é baseada em torno do conceito de "necessidades e capacidades", onde a SOA proporciona um mecanismo de ajustar as necessidades dos consumidores de serviços às capacidades dos fornecedores de serviços.

A eficiência no desenho, implementação e utilização dos sistemas baseados em SOA permite que uma organização se adapte muito mais rapidamente às alterações de negócio que ocorram.

Actualmente, o princípio das SOAs é considerado a melhor prática para o desenho e implementação de sistemas de informação [13] e, consequentemente, a Novabase decidiu propor à ABC a introdução gradual de Serviços e a conversão faseada da arquitectura actual dos sistemas de informação para uma SOA. Esta proposta, que foi bem recebida pela ABC, tem como objectivo orientar os sistemas de informação da ABC para o suporte dos processos de negócio existentes, dotando-os da flexibilidade e do conjunto de vantagens referidas anteriormente.

Desta forma, a arquitectura da aplicação a ser desenvolvida pode ser representada pelo esquema da figura 4.1 e da qual se passam a descrever os componentes mais relevantes:

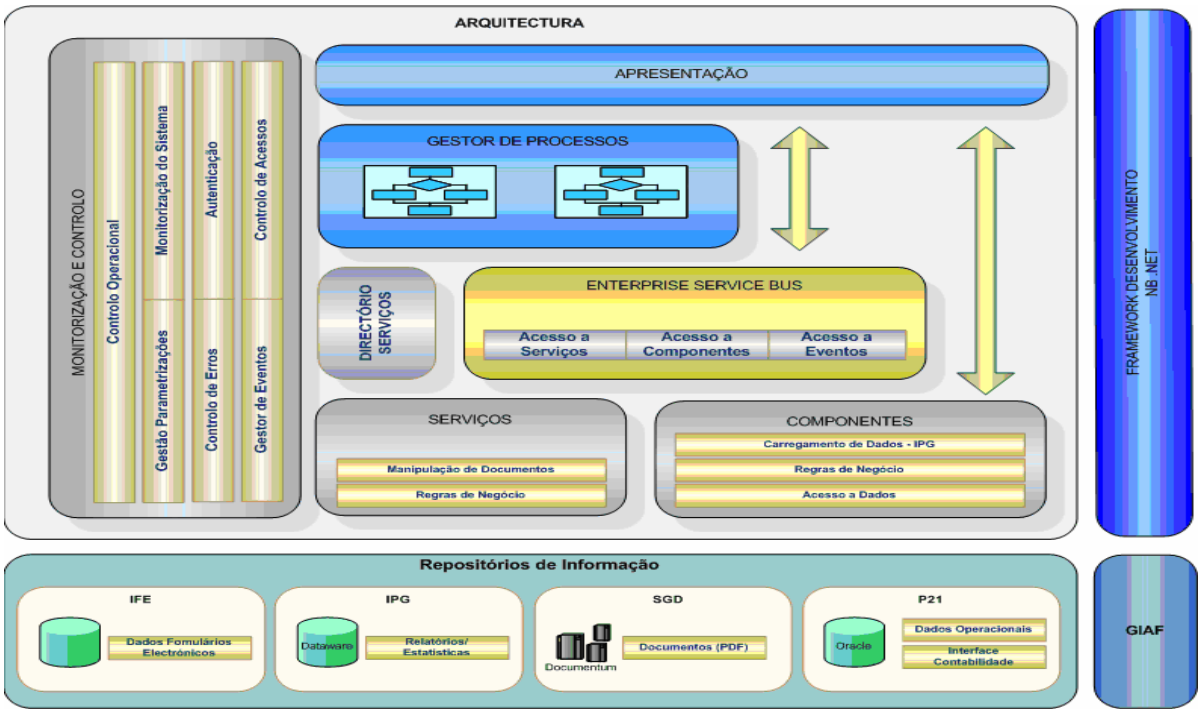

Figura 4.1 - Arquitectura SOA das aplicações da ABC

Camada de Apresentação

Esta camada é responsável por disponibilizar ao utilizador a informação de negócio que é processada nas camadas seguintes. Sempre que seja necessária a resposta do utilizador é esta a camada responsável pela sua recolha e encaminhamento.

Gestor de Processos

É a camada responsável pela implementação na SOA dos processos de negócio definidos através da utilização dos Serviços existentes, organizados de acordo com as regras de negócio do processo.

O gestor de processos será implementado através de *Workflows* administrados pela ferramenta *Oracle Workflow*.

Enterprise Service Bus

Esta camada é composta por um componente de alto desempenho que serve de interface entre os processos cliente e os serviços existentes, disponibilizando assim funcionalidades genéricas e reutilizáveis.

Permite que os clientes se foquem principalmente nas regras de negócio uma vez que fornece mecanismos *standard* para invocar serviços encapsulando também os detalhes de implementação do serviço.

Directório de Serviços

É a entidade que disponibiliza aos clientes a interface de todos os serviços disponíveis que fornecem funções de negócio relevantes para serem utilizados pela organização.

Serviços

A camada Serviços disponibiliza as funcionalidades de negócio utilizadas pelos processos sob a forma de serviços.

Um serviço pode ser o encapsulamento de componentes já existentes ou, sempre que se justificar, novos componentes desenvolvidos para uma arquitectura orientada a serviços.

Monitorização e Controlo

Disponibiliza o acesso a toda a informação relativa a Qualidade do Serviço (QoS) e também aos Níveis de Serviço contratados (SLA);

É a partir desta camada que será possível obter uma visão global sobre os diversos processos tratados na ABC e também obter informação detalhada sobre determinada instância de um processo em curso ou já terminado.

Contém o Gestor de Eventos, que é a camada responsável por gerir todos os eventos externos que são produzidos e que não são do controlo directo dos processos nem dos serviços.

Componentes

O actual modelo das aplicações (cliente/servidor) nem sempre garante o isolamento necessário entre as várias camadas. As aplicações contêm funcionalidades que passam pela apresentação, regras de negócio e acesso aos diversos componentes.

O objectivo é diluir progressivamente cada componente na respectiva camada chegando assim a uma única camada de apresentação que centraliza todas as funcionalidades da aplicação existente [14] convertendo os componentes em serviços. Esses componentes são:

- Ecrãs C++ ;

- Procedimentos PL/SQL;

- *Forms Oracle* (DIV, CTB).

Repositório de Informação

Corresponde aos diversos repositórios de dados utilizados pela aplicação

4.2. *FrameworkNB*

A *Framework* de desenvolvimento criada pela Novabase, conhecida como *FrameworkNB*, é baseada na *Framework .Net* 2.0 da Microsoft. A necessidade do seu desenvolvimento surgiu do facto de a maioria das aplicações de negócio serem muito complexas e os prazos serem normalmente muito apertados. Para além disso, equipas diferentes a trabalharem com a mesma tecnologia produziam código muito heterogéneo, para as mesmas funcionalidades.

Desta forma, a *FrameworkNB* permite:

- Homogeneidade entre projectos na mesma tecnologia;

- Simplificar a criação e manuseamento de entidades de negócio;

- Facilitar o desenvolvimento de ecrãs;

- Flexibilizar o acesso ao suporte relacional da aplicação (Base de Dados);

- Incorporar as melhores práticas de desenvolvimento nas aplicações a desenvolver.

Para concretizar este último ponto, a *Framework NB* tem como *design pattern* o modelo *Model View Controller*. Este padrão foi descrito em 1979 por Trygve Reenskaug [15].

4.2.1. *Model View Controller* (MVC)

No desenvolvimento de aplicações complexas há, muitas vezes, a necessidade de separar os dados da interface, de forma a que as alterações nesta não provoquem impacto nos dados e que estes possam ser reorganizados sem alterar a interface do utilizador. Este modelo resolve o problema separando as duas camadas referidas e introduzindo uma camada intermédia, o *Controller*.

No paradigma MVC as entradas do utilizador, a modelação do mundo externo e o *feedback* visual para o utilizador são explicitamente separados e tratados por três tipos de objectos, cada um especializado na sua tarefa.

Como se pode ver na figura 4.2, a *View* é a interface com o utilizador e deve apenas preocupar-se com essa interacção.

O *Controller* é quem "gere" a ligação entre o *Model* e a *View*, tratando de todas as interacções entre o utilizador (através do Interface) e o modelo de negócio. Interpreta o *input* do rato e do teclado, dizendo ao *Model* ou à *View* para se alterarem de acordo com esse *input*.

Finalmente, o *Model* representa o modelo de negócio e descreve todas as regras e informação que compõem o domínio da aplicação, gere o comportamento e os dados desse domínio e responde a pedidos de alteração de estado (normalmente, vindos do *Controller*).

A separação formal destas três tarefas é uma noção importante, em que o comportamento básico pode ser incorporado em objectos abstractos: *View*, *Controller e Model* [16].

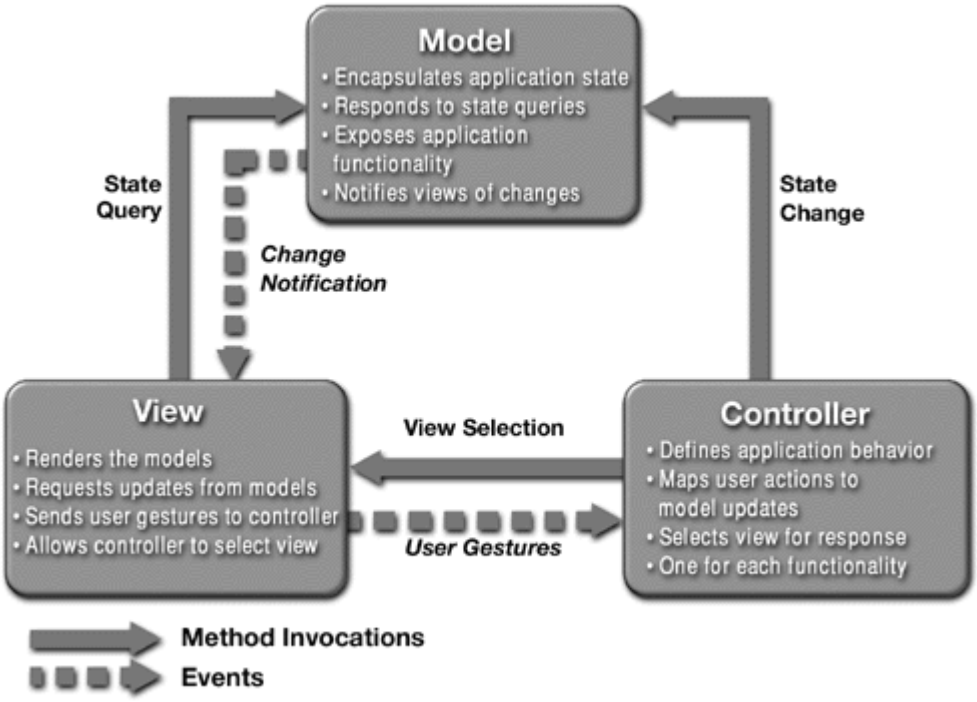

Figura 4.2 – A arquitectura Model-View-Controller

É importante notar que tanto o *View* como o *Controller* dependem do *Model*. No entanto, o modelo não depende de nenhum dos dois. Este é um dos benefícios chave da separação. Esta separação permite que o modelo seja construído e testado independente da apresentação visual. A separação entre o *View* e o *Controller* é secundária em muitas aplicações *fat-client,* como por exemplo *Windows Forms* e, de facto, muitas *frameworks* de interface de utilizador representam os dois papéis como sendo um objecto [17].

Modelo MVC em *Windows Forms*

Em *Windows Forms*, as práticas para o *View* e para o *Controller* estão muito bem definidas. É deixado ao programador o desenho do *Model*.

Model

Em *Windows Forms*, não é estritamente necessário que exista um *Model*. O programador tem a opção de criar uma classe de modelo mas, em vez disso, pode preferir ter os gestores de eventos no *Controller* a fazer cálculos e persistência de dados. Mais uma vez, usar um modelo para encapsular regras de negócio e acesso a base de dados é tanto possível como preferível. É deixado aos programadores desenharem esse modelo.

View

Uma classe que herde tanto de *Form* ou de *Control* detém as responsabilidades da *View*. É tipicamente o ecrã ou o controlo que serve de *interface* para o utilizador poder visualizar e manipular dados.

Controller

Os deveres do *Controller* estão divididos em três locais. O gerar e passar eventos é realizado ao nível do Sistema Operativo. Dentro da *Framework .Net*, as classes *Form* e *Control* encaminham o evento para o gestor de eventos apropriado. A gestão de eventos é normalmente realizada na classe de código e pode ser alterada [18].

4.2.2. Arquitectura da *FrameworkNB*

O desenvolvimento deste projecto assenta na arquitectura representada na figura 4.3.

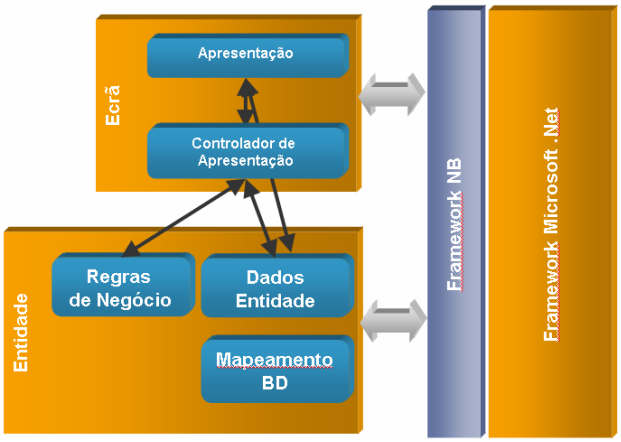

Figura 4.3 - Arquitectura da *FrameworkNB*

Área de Ecrã:

Na área de ecrã, encontra-se a camada de **Apresentação**, que consiste na própria Interface/Ecrãs Base que compõem a aplicação. Estes não contêm o desenho específico dos ecrãs. Servem apenas para disponibilizar um conjunto de serviços da infra-estrutura base para os mesmos.

Os ecrãs base disponibilizam informação de contexto onde o ecrã está inserido, nomeadamente:

- Actividade/Tarefa de Processo;

- Registo actual;

- Transacção da Base de Dados.

Todas as acções a serem efectuadas por esta camada devem delegar o comportamento para o **Controlador de Apresentação**. Este controlador disponibiliza os serviços necessários à implementação do ecrã, sejam eles apenas regras de interface ou operações a serem realizadas pelo Modelo de Classes. É também responsável por fornecer os serviços de navegação e manutenção de contexto (se o ecrã o suportar).

Área de Entidade

É na camada de **Regras de Negócio** que se encontra a Classe de Entidade, que contém todas as regras de negócio e métodos da Entidade a implementar, tais como descritos no Modelo de Classes. Para além disso, comporta todos os métodos de persistência de dados da mesma, pois tem uma referência para o tradutor de Modelo Físico/Lógico.

A camada de **Dados da Entidade** representa o próprio *DataSet*. O *DataSet* é uma estrutura da *Framework .Net* para guardar dados em memória. É uma representação em memória de uma colecção de objectos de base de dados, incluindo tabelas de um esquema de base de dados relacional. O *DataSet* pode ser utilizado para manipular dados localmente e no final sincronizá-los com a base de dados física. Desta forma, possibilita formas desconectadas de trabalhar com dados, o que melhora o desempenho, pois reduz o número de vezes que a base de dados é acedida para manipulações de dados [19].

O *DataSet* contém todos os objectos comuns a uma *Collection*. Contém as tabelas da base de dados como objectos *DataTable*, relações como *DataRelations* e restrições como *Constraints* e as relações entre estes vários objectos podem ser observadas na figura 4.4. Os dados da base de dados são preenchidos no *DataSet* através do *DataAdapter*.

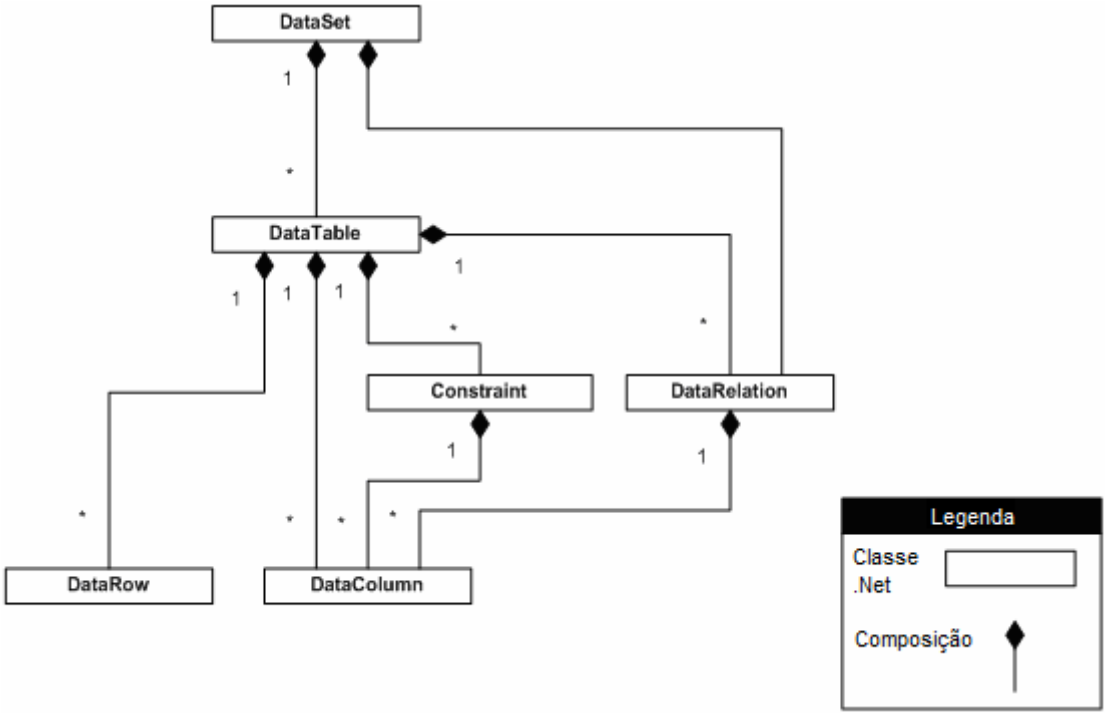

Figura 4.4 - Hierarquia do *DataSet*

O *DataSet* é utilizado neste projecto como o esquema da informação da entidade, exactamente como especificado no Modelo Lógico de Dados. Serve também como contentor dos dados da entidade que vão ser manipulados pela aplicação (quer na interacção com o utilizador através do ecrã, quer através dos processamentos específicos dessa entidade).

O **Mapeamento BD** permite configurar todas as instruções SQL que vão traduzir os dados físicos de uma entidade em dados lógicos, através de um ficheiro XML. É utilizado pelo Tradutor Modelo Físico/Lógico, denominado *DacManager*, que é disponibilizado às Classes de Entidade para "carregar" os dados físicos da entidade no Modelo Lógico. Este tradutor permite efectuar, através de configuração, o mapeamento entre o Modelo Físico de BD (tabelas de Base de Dados) e o Modelo Lógico de BD (*DataSets* de Entidades).

4.3. Regras de Construção de Ecrãs

Existe um conjunto de regras que enquadram o modo de criação dos ecrãs do SiABC com base nas características dos modelos conceptuais do negócio da ABC, especificados de acordo com a metodologia de desenvolvimento definida.

O objectivo é, de uma forma clara, definir quando e como se deve criar um ecrã no novo sistema, como relacionar vários ecrãs num agregado e quando invocar outros agregados [1].

Pretende-se normalizar a visão e a manipulação dos conceitos, ou seja, independentemente do contexto de utilização, a visualização dos dados de um determinado conceito é constante. As acções possíveis sobre um determinado conceito são igualmente constantes, variando somente o conjunto de permissões atribuídas ao utilizador de acordo com o seu perfil e o contexto de utilização, ou seja, o processo em execução.

Da mesma forma, pretende-se que os processos de negócio sejam realizados sempre da mesma maneira, o que significa normalizar a execução das respectivas acções. Para tal, a visão global dos dados de um processo e a visão específica dos dados de determinada acção desse processo deve ser única, independentemente do contexto em que o processo é executado. À semelhança do que é pretendido para os conceitos, o conjunto de acções possíveis e de dados visíveis, em determinado momento, para cada utilizador, dependerá das permissões do mesmo e do contexto em que o processo é executado.

Na prática, isto significa que para cada modelo conceptual, é construído um conjunto de ecrãs único que é reutilizado em todas as situações em que é necessário visualizar ou manipular o conceito ou processo associado ao referido modelo conceptual.

Em cada ecrã, como representado na figura 4.5, os objectos de dados correspondem aos atributos de cada classe e são armazenados no modelo de dados lógico. As acções do ecrã invocam as operações das classes.

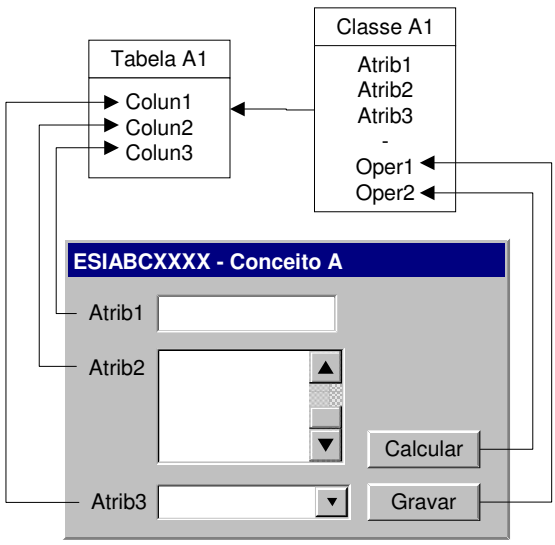

Figura 4.5 - Esquema de ligação do ecrã à base de dados

No caso dos conceitos, o conjunto de ecrãs associado ao respectivo modelo conceptual é composto por um ecrã com os dados e acções das classes próprias do modelo e por um ecrã por cada associação a conceitos externos ao modelo.

Para melhorar a utilidade e contextualizar os ecrãs que representam as relações com os conceitos possíveis, os mesmos deverão incluir dados do referido conceito externo. No entanto, estes ecrãs não permitem a execução de acções sobre os referidos conceitos externos. Essas acções estão reservadas aos ecrãs associados ao modelo conceptual dos respectivos conceitos externos.

No ecrã associado aos conceitos internos do modelo conceptual, cada classe é apresentada num separador próprio, como se pode ver na figura 4.6.

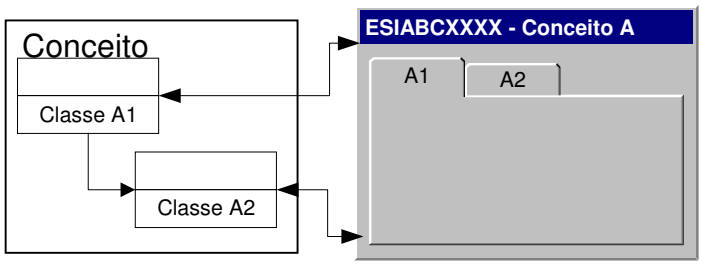

Figura 4.6 - Demonstração da utilização de separadores

O mesmo tipo de separação é utilizado quando a representação gráfica dos atributos de uma única classe não é possível na área útil de um separador de ecrã.

Nesse caso, é aplicado um critério funcional, de acordo com cada conceito, para distribuir os atributos de uma classe por vários separadores.

Para os processos foi também definido um conjunto de regras de criação de ecrãs, mas a fase de desenvolvimento de processos não se enquadra no âmbito deste relatório.

Agregado

Um agregado é um conjunto de ecrãs existentes, agrupados de acordo com um critério funcional, com o objectivo de permitir ao utilizador realizar no sistema de informação as tarefas que lhe competem.

A ligação dos ecrãs é efectuada através de abas. Cada aba corresponde a um ecrã e o seu esquema pode ser visualizado na figura 4.7.

Figura 4.7 - Demonstração da utilização de um agregado

É possível criar agregados compostos apenas por um ecrã. Nesse caso, graficamente, não será visível a aba associada ao ecrã.

Para os conceitos, por norma, será sempre criado um agregado base que garanta a possibilidade de executar as tarefas de manutenção do ciclo de vida do conceito: Criação, Alteração, Consulta e Eliminação. Este agregado é composto pela totalidade dos ecrãs criados para o modelo conceptual associado ao conceito em causa.

Para além do agregado base, podem ser criados, para um determinado conceito, mais agregados caso, no âmbito de um processo de negócio implementado no sistema através de *Workflow*, seja necessário agir directamente sobre o conceito de uma forma para além das acções previstas no respectivo agregado base. Da mesma forma, se as novas acções sobre o conceito, não estiverem enquadradas num processo de negócio existente, ou seja, acções *ad-hoc*, poderão ser criados agregados específicos para satisfazer essas necessidades.

Em qualquer agregado, existe o conceito de ecrã principal ou aba principal. O ecrã principal corresponde ao modelo conceptual que está a ser manipulado pela tarefa em execução. As acções aplicáveis ao agregado, nomeadamente, gravação da informação dos vários ecrãs, confirmação da execução da tarefa, cancelamento da tarefa e fecho do agregado (de todos os ecrãs), apenas estarão disponíveis no ecrã principal. O ecrã principal é sempre o primeiro ecrã visível do agregado.

Como referido anteriormente, cada agregado e, consequentemente, as acções disponíveis nesse agregado só dizem respeito a um determinado conceito ou processo. Se for relevante, no decorrer da utilização de um agregado, manipular outro conceito que não o principal do agregado em questão, haverá várias hipóteses:

- A manipulação do outro conceito é uma tarefa intrínseca ao conceito do agregado em utilização e ocorre frequentemente. Neste caso, poderão ser adicionadas ao agregado novas abas que permitam a manipulação de um outro conceito, sem haver a necessidade de utilizar um agregado associado a esse outro conceito, como ilustrado na figura 4.8;

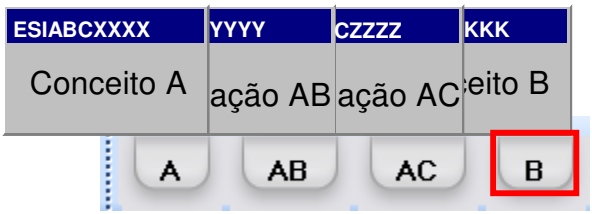

Figura 4.8 - Caso em que o novo conceito é utilizado frequentemente

- A manipulação do outro conceito é uma tarefa esporádica e o trabalho realizado no conceito base não pode ser perdido. Neste caso, a partir da aba relevante no agregado base, deverá haver uma acção que permita invocar o agregado associado ao outro conceito a manipular e deverá ser garantido que, ao terminar a manipulação do outro conceito, volta-se ao agregado original, como representado na figura 4.9;

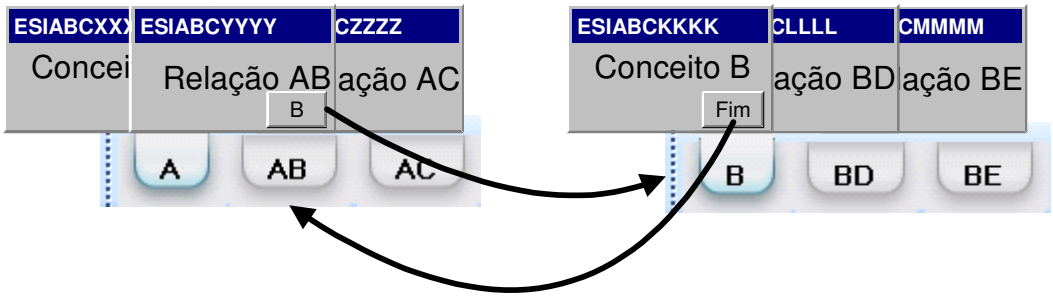

Figura 4.9 - Caso em que o novo conceito é utilizado esporadicamente, mas sem perder os dados do principal

- A manipulação de um outro conceito é esporádica e as acçoes da tarefa em execução podem ser perdidas. Neste caso, ilustrado na figura 4.10, a utilização do agregado base pode ser concluída ou cancelada conforme a decisão do utilizador e a manipulação do outro conceito será feita através do agregado a ele associado, invocado através de uma opção de menu.

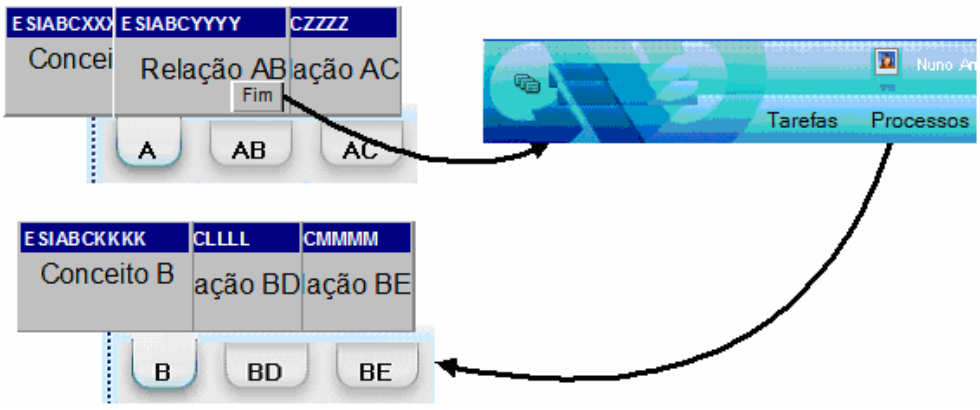

Figura 4.10 - Caso em que o novo conceito é utilizado esporadicamente, com perda dos dados do principal

Capítulo 5

5. Análise e Desenvolvimento

5.1. Tarefas iniciais

No momento em que fui colocada no cliente iniciei uma formação de duas semanas que consistia em ler documentação e fazer exercícios que envolviam a *FrameworkNB* a *Framework .Net* 2.0, com a linguagem C# e a Base de Dados *SQL Server*. Estas semanas foram muito importantes para eu tomar conhecimento da estrutura da *framework* que iria trabalhar futuramente, como se organizava e quais as suas principais classes e funcionalidades.

Desta forma, o meu trabalho inicial incidiu na construção da *FrameworkABC*. Esta consiste numa plataforma assente na *FrameworkNB*, com a adição de um conjunto de classes que permitirá que o desenvolvimento dos vários módulos seja feito de uma forma rápida e consistente com todos. Esta infra-estrutura também engloba as funcionalidades comuns a todos os módulos, tais como:

- Mensagens de Sistema (de erro, informação, etc.);

- Parâmetros de Referência (necessários para as parametrizações dos vários conceitos);

- Entidades (Utente, Serviços, Entidades Externas).

Na altura em que iniciei o meu trabalho, a *FrameworkNB* já estava instalada na ABC, e alguns componentes ABC já tinham sido desenvolvidos. Assim, comecei por desenvolver alguns ecrãs básicos para me ir ambientando com a maneira de funcionar com a *FrameworkNB*.

5.2. Gestão de Nacionalidades

Consiste num simples ecrã de parametrização que permite visualizar, modificar, inserir e apagar nacionalidades, para que possam ser utilizadas pelo sistema, tal como se ilustra na figura 5.1.

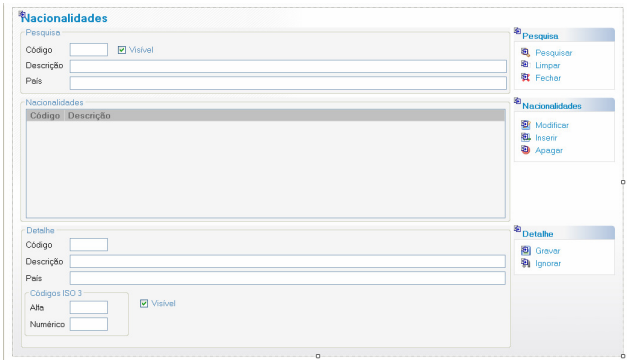

Figura 5.1 - Ecrã de Gestão das Nacionalidades

5.3. Ecrãs Base

De forma a facilitar o desenvolvimento e a não repetição de código e de funcionalidades comuns a vários ecrãs, foram criadas 5 classes "pai" de todos os ecrãs. Apesar de estas classes já estarem construídas quando iniciei o projecto, tiveram de sofrer muitas alterações à medida que íamos percebendo as necessidades do cliente e acabei por dar grande contributo para a versão final das mesmas. A hierarquia das classes pode ser observada na figura 5.2 e segue-se uma breve descrição das mesmas.

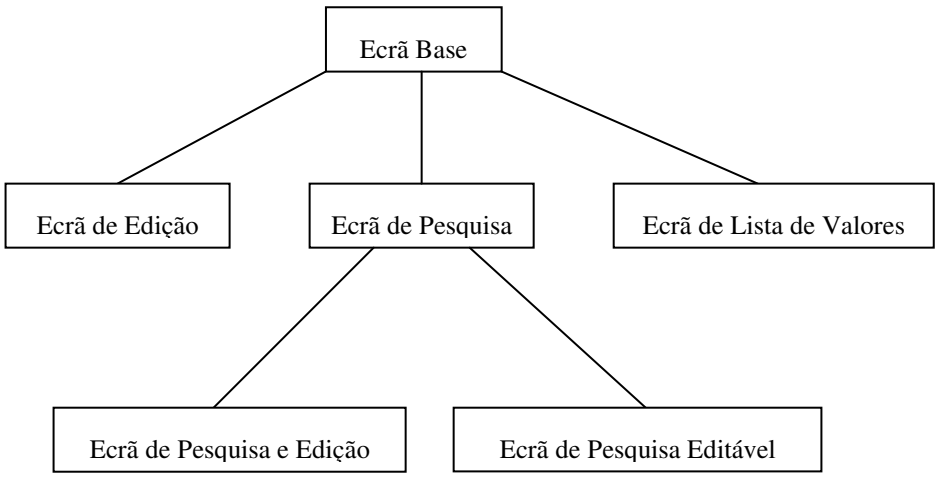

Figura 5.2 - Hierarquia das classes de Ecrãs

Ecrã Base

Este ecrã é herdado por todos os outros. Pode servir de base para qualquer formulário. Não implementa nenhuma acção, mas tem métodos para apresentar o nome do ecrã na barra de janelas da aplicação.

Ecrã de Edição

O ecrã de edição suporta as funções de Gravar e Cancelar. Quando o utilizador clica em "Gravar", esta classe é responsável por passar os dados do Ecrã para o *DataSet*, e deste para a base de dados. O *layout* dos ecrãs base de pesquisa, que serão herdados por outros, pode ser visualizado na figura 5.4. E como se pode verificar na figura 5.3, os ecrãs que herdam deste também podem ter as suas próprias acções.

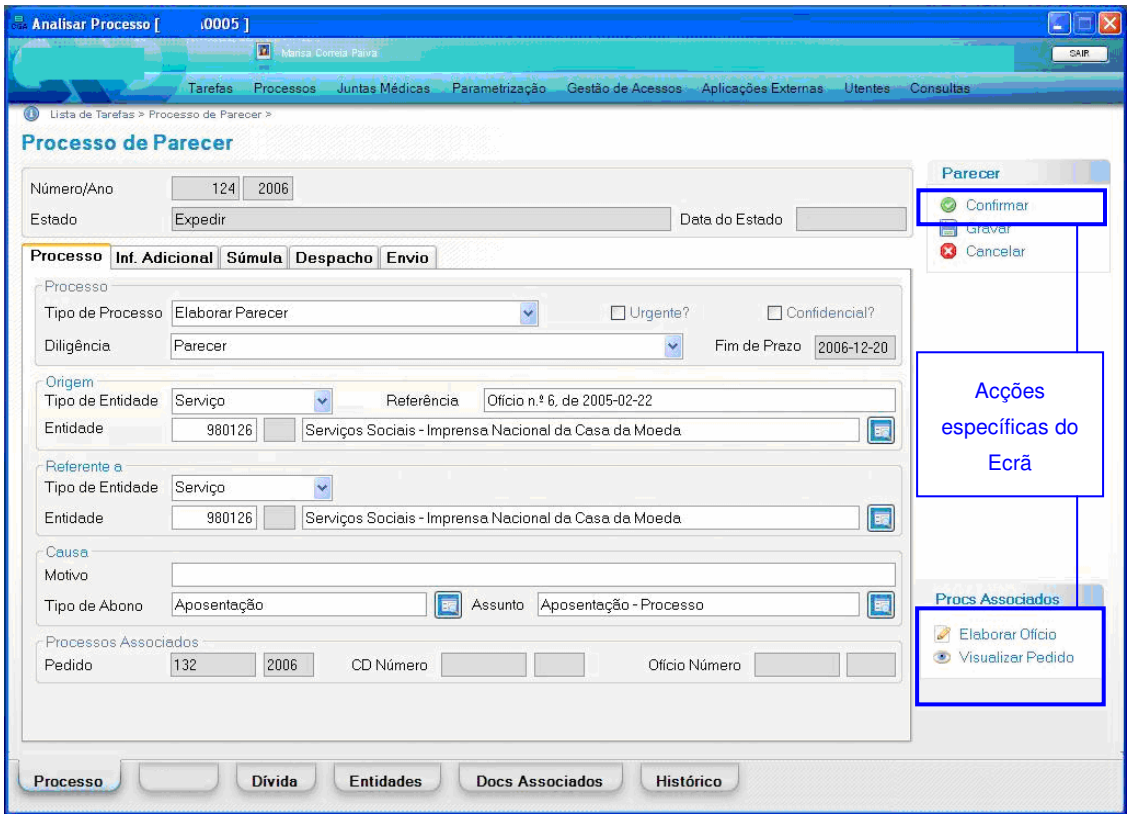

Figura 5.3 - Ecrã herdado do Ecrã de Edição

Ecrã de Pesquisa

Este ecrã apresenta uma área de Pesquisa e uma área para a lista de resultados. Tem as funções de pesquisa na base de dados sobre uma entidade e já tem as acções necessárias para chamar o ecrã que irá permitir visualizar ou modificar entidades existentes e criar novas entidades.

Ecrã de Pesquisa e Edição

Este ecrã herda do Ecrã de Pesquisa e, dessa forma, é também constituído por uma área de Pesquisa e outra para a lista de resultados. Para além dessas, tem uma área de detalhe em que é possível o utilizador modificar e apagar registos existentes na base de dados e inserir novos registos do conceito em questão, sem sair do ecrã.

Ecrã de Pesquisa Editável

Ecrã que contém as mesmas áreas que o Ecrã de Pesquisa, mas contém as funções de modificar, inserir, apagar e gravar o *Dataset* e o registo da própria base de dados na própria "Grelha de Resultados".

Ecrã de Lista de Valores

Este ecrã é normalmente aberto a partir de um botão de *QuickList* e apresenta a lista de todos os registos existentes de um determinado conceito, também com uma área de Pesquisa. Serve para poder ser facilmente encontrado o registo que o utilizador procura, normalmente para preencher um campo de outra pesquisa.

Áreas de trabalho nos ecrãs

Para manter a coerência e a usabilidade dos ecrãs da aplicação SiABC, estes são constituídos por diversas zonas de funcionalidade. Essas áreas, que se descrevem sumariamente em seguida, exemplificam-se nas figuras 5.4 e 5.5 e podem variar mediante o objectivo e o contexto do ecrã:

- Área de Pesquisa - Zona do ecrã onde é possível preencher os filtros de pesquisa específicos de um contexto;
- Grelha de Resultados - Lista de registos resultantes da pesquisa, tendo em conta os filtros inseridos;
- Área de Trabalho - Área onde o utilizador pode consultar e gerir informação;
- Área de Acções - Área que permite efectuar determinadas acções sobre a informação visualizada;
- Área de Contexto - Área reservada a permitir visualizar a informação que provém de outro ecrã num contexto específico;

- Separadores - Quando a informação a gerir num ecrã é volumosa ou complexa, pode ser dividida em diversos Separadores (ver explicação no Capítulo 4.3);
- Agregado - Agrupamento de vários ecrãs existentes, ligados através de abas (ver explicação na Secção 4.3).

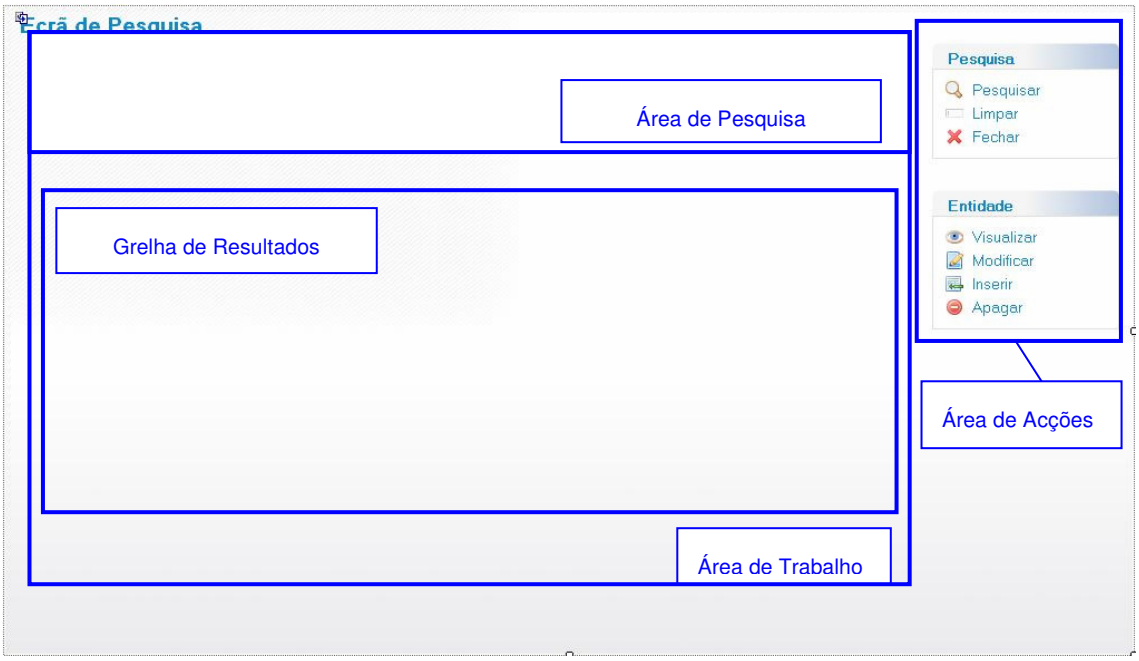

Figura 5.4 - Ecrã Base de Pesquisa, com identificação das áreas de trabalho

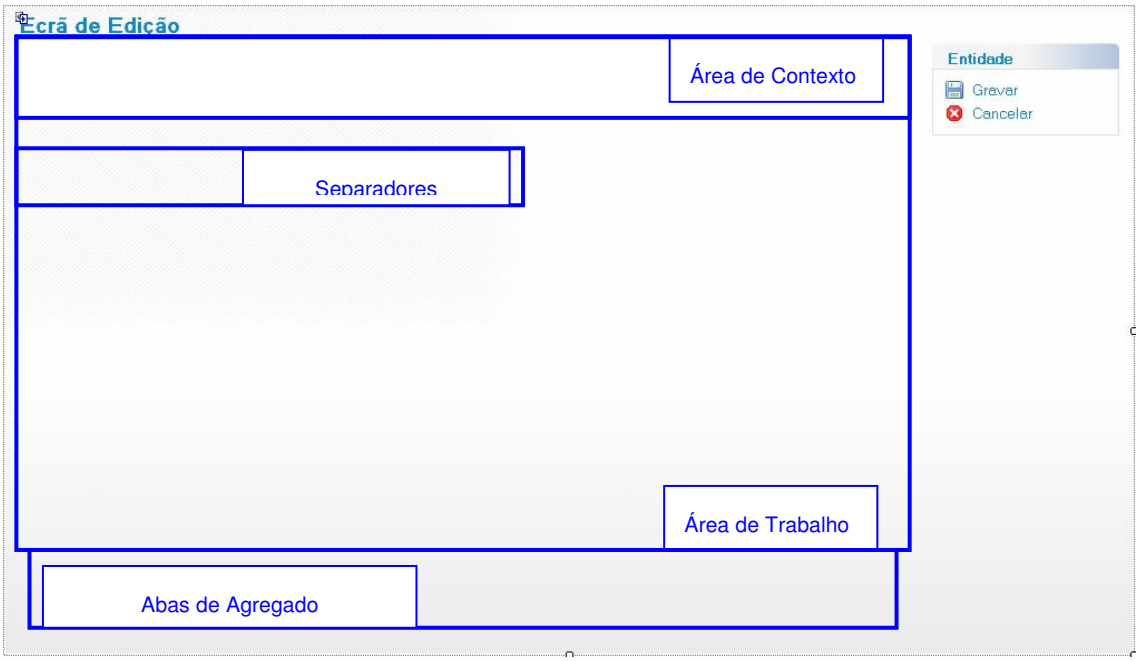

Figura 5.5 - Ecrã Base de Edição, com identificação das áreas de trabalho

5.4. Controlo para os Perfis

Na linha de funcionamento dos ecrãs de parametrização, fez-se sentir a necessidade de um ecrã que permitisse ao utilizador associar regras aos perfis existentes na aplicação e remover associações existentes. Assim, criou-se uma nova funcionalidade (que mais tarde foi adaptada para ser genérica e se tornar num componente ABC) que se comportasse como um controlo: *Add-and-Remove List Boxes*, cujo resultado pode ser visto na figura 5.6.

Figura 5.6 - Controlo Add-and-Remove List Boxes

5.5. Módulo de Consultas

Sendo a aplicação um Sistema de Informação, deverá ter uma grande preocupação em facilitar ao máximo aos utilizadores a procura da informação que desejam. Sendo assim, verificou-se a necessidade de haver uma forma de os utilizadores poderem abrir várias janelas ao mesmo tempo com diferentes pesquisas. Para resolver este requisito foi criado o Módulo de Consultas. Este módulo consiste num conjunto de ecrãs, acessíveis a partir da opção de menu Consultas, que são abertos apenas em Modo View (não permitem edição) e num outro ecrã no qual o utilizador pode definir qual é a Entidade que deseja procurar, isto é, o Contexto. Este contexto será a entidade que aparecerá por omissão nos campos de pesquisa de todos os ecrãs abertos neste Módulo.

De forma a possibilitar um acesso mais rápido e sempre disponível a este Módulo, quando se abre a aplicação aparecerá um ícone na área de Notificação do Windows (canto inferior direito), como se pode ver na figura 5.7, que terá as mesmas funções que a opção de Menu Consultas.

Figura 5.7 - Ícone do Módulo de Consultas

À medida que se vão abrindo os vários ecrãs de Consulta, como representado na figura 5.8, o título do respectivo ecrã é acrescentado ao Menu do Módulo de Consultas, de forma a poderem ser facilmente acedidos.

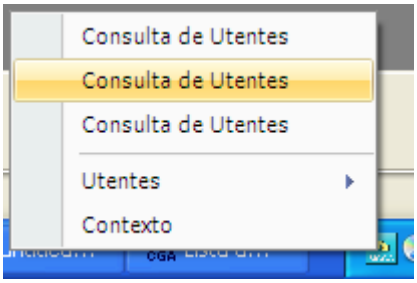

Figura 5.8 - Menu do Módulo de Consultas, com 3 ecrãs de Consulta de Utentes abertos

Foi sobre este Módulo de Consultas que fiz toda a análise e desenho.

5.5.1. Módulo de Consultas – Levantamento de Requisitos

A partir de conversas com o cliente e de documentação fornecida pela Novabase sobre reuniões anteriores com o mesmo, reconheci os seguintes requisitos para este módulo, como se pode ver na figura 5.9.

Requisitos Funcionais

- Módulo de Consultas:

O Módulo de Consultas é um conjunto de ecrãs que são abertos em Modo de Consulta, ou seja, apenas para visualização. Tem como objectivo permitir a consulta de diferente informação em simultâneo, pois podem ser abertos vários ecrãs ao mesmo tempo. Permite a definição de um contexto de consulta.

- Definir Contexto:

Permitir indicar um contexto explícito. Este contexto consiste nos dados da entidade a consultar, que é utilizado na abertura dos vários ecrãs do Módulo de Consultas.

- Consultar:

Abrir os ecrãs com os dados do contexto. Se não for indicado contexto, as consultas incidirão sobre a última entidade consultada.

Quando é o primeiro ecrã a ser aberto no Módulo de Consultas e não existe contexto, os campos deverão estar vazios.

- Acesso Ecrãs Abertos:

Possibilitar um acesso rápido e fácil aos ecrãs abertos pelo Módulo de Consultas.

- Ícone Módulo de Consultas:

Possibilitar o acesso ao Módulo de Consultas através de um ícone na área de Notificação do Windows. Este deverá conter as mesmas funções que a opção do menu que abrirá o Módulo de Consultas.

Requisitos Não Funcionais

- Reutilização de Ecrãs:

Os ecrãs que serão utilizados para o Módulo de Consultas deverão ser os mesmos criados para o Módulo de Operação, mas em Modo de Consulta.

- Limite de Ecrãs Abertos:

Limitar o número de Ecrãs abertos pelo Módulo de Consultas.

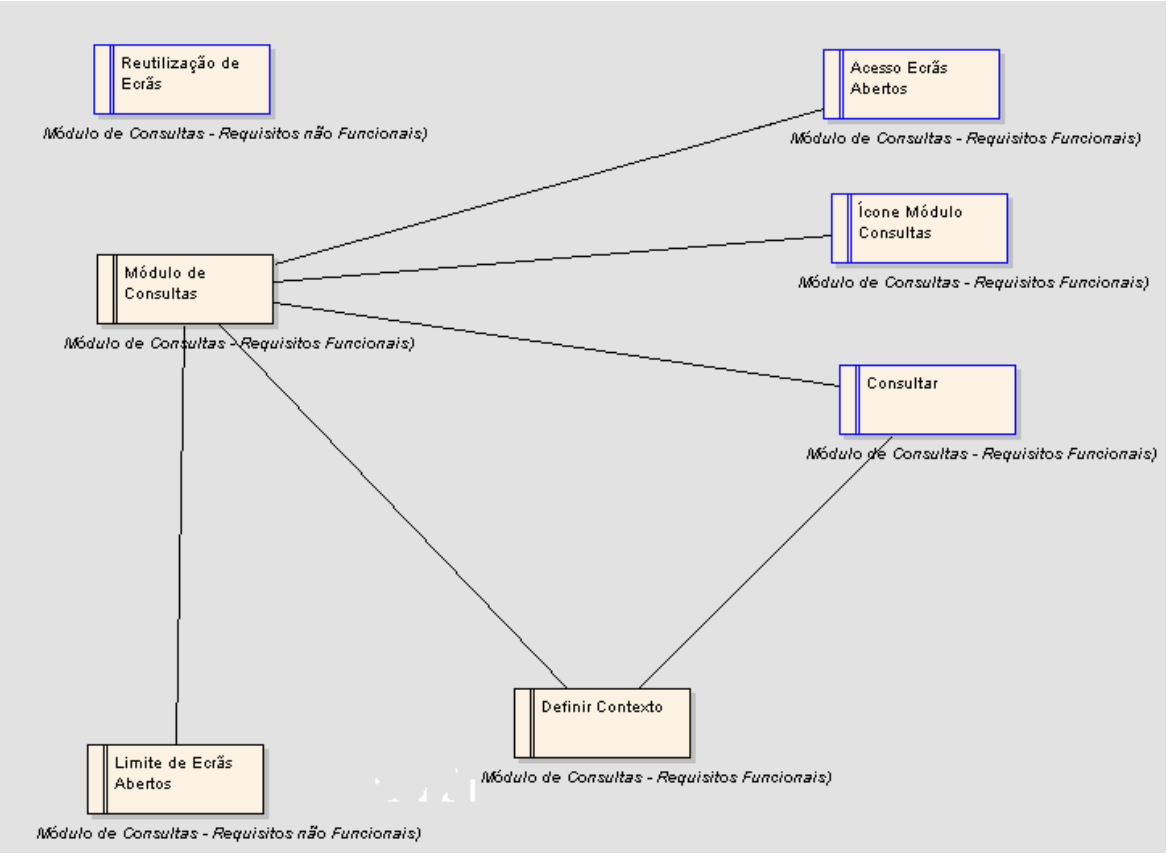

Figura 5.9 - Levantamento de Requisitos do Módulo de Consultas

5.5.2. Módulo de Consultas – Casos de Uso

Identifiquei os seguintes Casos de Uso, representados na figura 5.10.

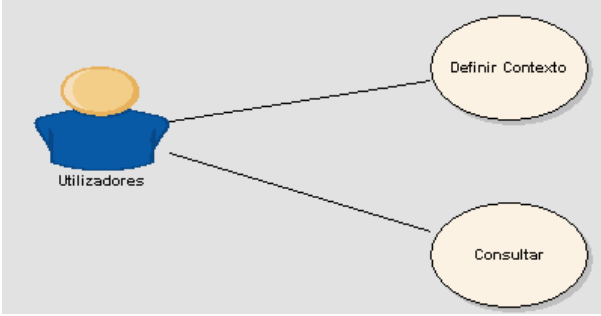

Figura 5.10 - Caso de Uso do Módulo de Consultas

Caso de Uso - Definir Contexto

Este Caso de Uso consiste em preencher os campos de pesquisa que servirá de Contexto para as outras consultas. Sendo assim, o Diagrama de Actividades será como está representado na figura 5.11 e segue-se uma breve descrição das suas actividades.

- Actividade Definir Contexto

Para definir um contexto, o utilizador deverá escolher o tipo de Entidade e preencher o Código da Entidade, podendo recorrer à ajuda da Lista de Valores que se deseja consultar. Após o "Confirmar", essa Entidade será o Contexto do Módulo de Consultas.

- Actividade Definir Contexto Vazio

Para eliminar o contexto existente, o utilizador deverá limpar os campos da Entidade. Após confirmar, o contexto estará vazio.

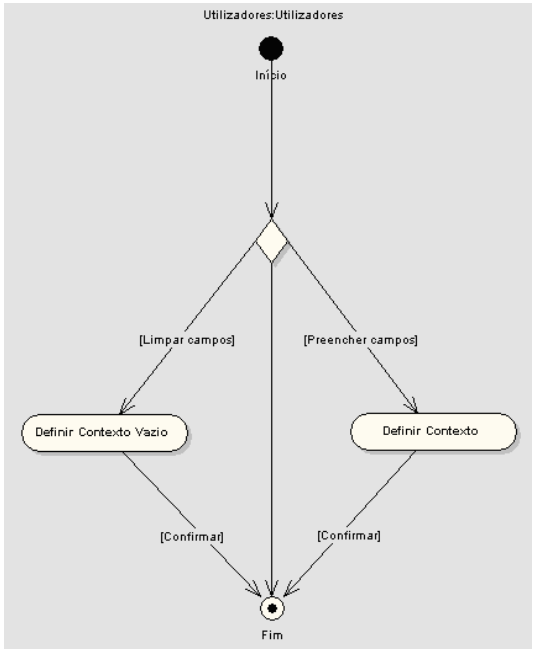

Figura 5.11 - Diagrama de Actividades do Caso de Uso: Definir Contexto

Caso de Uso: Consultar

Abrir ecrãs do Módulo de Consulta. Desta forma, o Diagrama de Actividades terá apenas a actividade em que o utilizador consulta a informação que deseja.

5.5.3. Módulo de Consultas – Modelo de Classes

Para implementar os casos de uso identificados, foi necessário criar uma série de classes, tal como se apresentam na figura 5.12, as quais se descreve sumariamente em seguida.

- Contexto:

Classe que pede ao utilizador os dados de contexto (Tipo Entidade, Código, Sufixo e Descrição) e os guarda num *DataSet*. Contém os métodos *Confirmar*(), e *GetSearchContext*(), que serão descritos mais à frente.

- *NBSearchEntityControllerHandler*:

Handler de Menu que abre o Ecrã de Contexto.

- *StaticSearchContextContainer*:

É o "repositório" do contexto. É a classe responsável por guardar uma *DataRow* com o contexto, através do método *GetSearchContext*() e passar o contexto quando é chamado, através do *SetSearchContext*(Entidade).

- *frmMDIHostForm*:

MdiForm que vai conter os *Forms* do Módulo de Operações abertos em Módulo de Consultas. Tal como o *StaticSearchContextContainer*, contém os métodos *GetSearchContext*() e *SetSearchContext*(Entidade), sendo o responsável por passar o contexto para os Ecrãs a abrir.

- *NBSearchStartControllerHandler*:

Handler de Menu que, através do método *GetHostingForm*(), sabe qual o ecrã do Módulo de Operações seleccionado pelo utilizador através do menu e abre-o, em Modo de Consulta.

- *NotifyIconManager*:

Trata das operações necessárias para que o *Icon* tenha o mesmo comportamento que a opção de Menu. Vai buscar os itens de menu através do método *InitializeShellData*(int, Ecra) e regista o menu pai, no método *InitializeChildWindowMenu*().

- *MenuChildWindowManager*:

Cria o Menu do *Icon*. Através dos métodos *RegisterChildWindow*(Ecra) e *CreateChildWindowItem*(Ecra), cria as opções de menu iniciais e, à medida que o utilizador vai abrindo novos ecrãs do Módulo de Consultas, esta classe cria as novas opções de menu, para que o resultado final seja como já foi apresentado na figura 5.8.

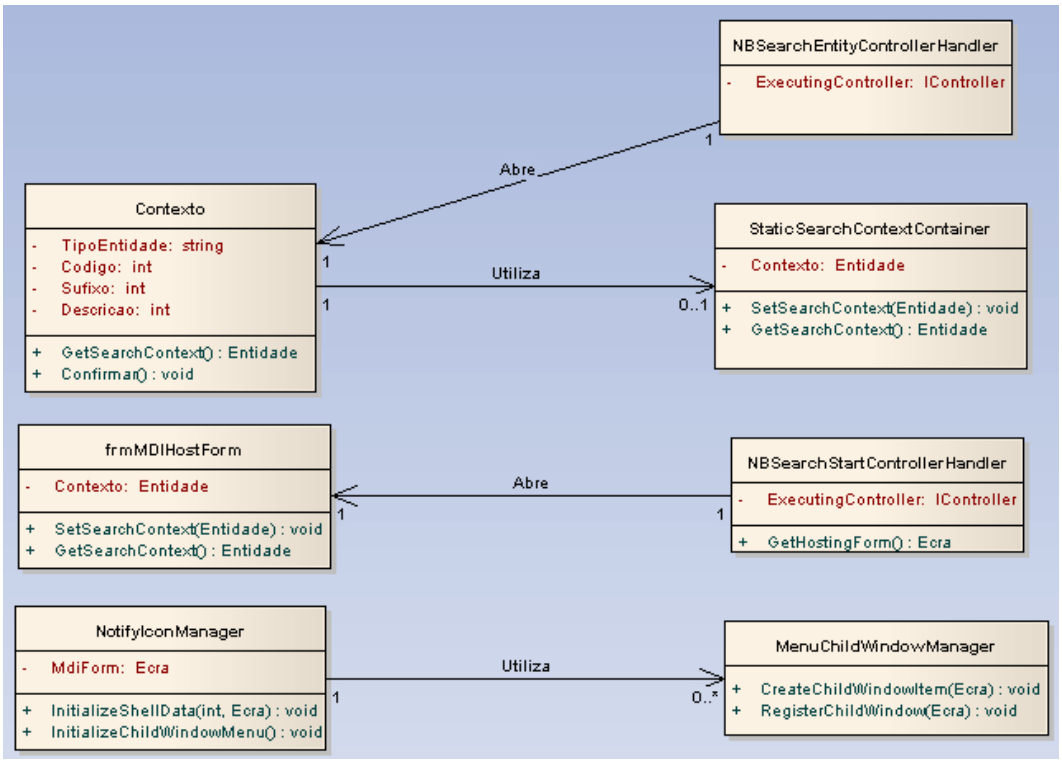

Figura 5.12 - Diagrama de Classes do Módulo de Consultas

5.5.4. Módulo de Consultas - Desenvolvimento

Para aceder ao Módulo de Consultas, o utilizador deverá utilizar a opção de Menu: Consultas (figura 5.13) ou, como já foi referido, o *icon* da área de Notificação.

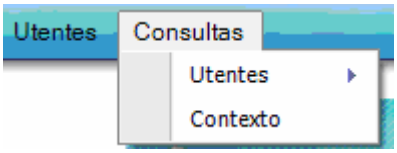

Figura 5.13 - Menu do Módulo de Consulta

A partir desde Menu, o utilizador pode "Definir um Contexto", que consiste em abrir o ecrã de Definição de Contexto (figura 5.14).

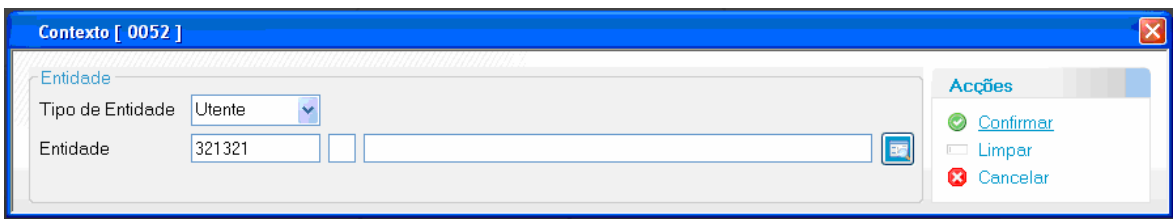

Figura 5.14 - Ecrã de Definição de Contexto

Aqui, o utilizador pode seleccionar uma entidade, com a ajuda da Lista de Valores.

Se fizer "Confirmar" com uma Entidade Preenchida, a classe Contexto procura esses dados com o dsContexo (*DataSet* de Contexto) e devolve um registo com todos os dados dessa entidade.

Seguidamente, pede ao seu *Controller* para definir o contexto, que por sua vez manda o *StaticSearchContextContainer* fazer o *SetSearchContext* com aquele registo. O método *SetSearchContext* passa o registo do contexto à variável Contexto do *StaticSearchContextContainer*. Após isso, o Ecrã de Definição de Contexto fecha.

A classe responsável pela abertura de um ecrã de consultas através do Menu (Consultas → Utentes → Consulta de Utentes) é a *NBSearchStartControllerHandler*. Esta passa o contexto que está "armazenado" no *StaticSearchContextContainer* para o *frmMdiHostForm* .

Os ecrãs do Módulo de Consultas são abertos por um *SearchContextController* que tem um método *InitializeForm()*. Este método tem um *receiver* (*Form* Consulta Utente) e um *donator* (*frmMdiHostForm*) de contexto e diz ao *receiver* para *set()* e ao *donator* para *get()*.

Quando o ecrã Consulta Utente abre, porque implementa a interface *ISearchContextReceiver,* o *frmMdiHostForm* passa-lhe o contexto. Desta forma, quando aparece no ecrã, já vem com os campos de pesquisa preenchidos com o contexto, como se pode ver na figura 5.15.

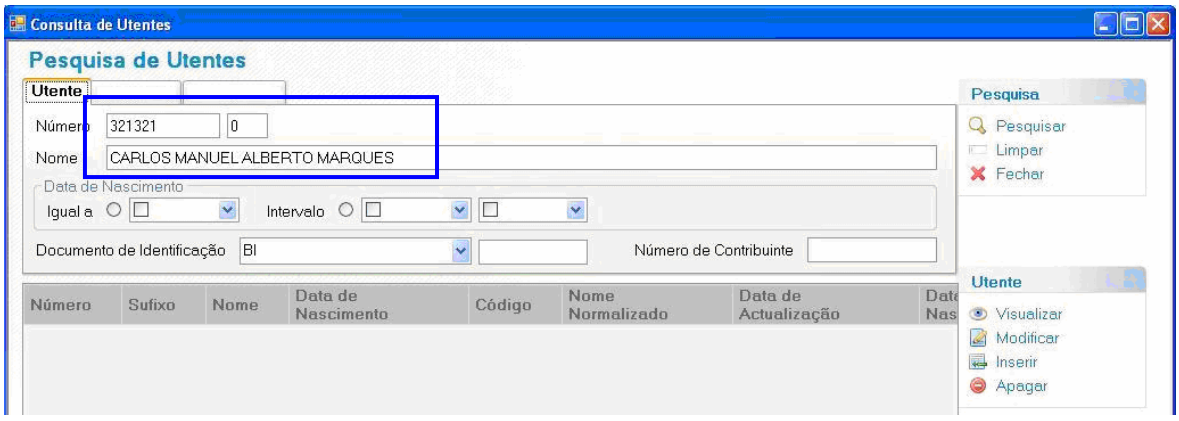

Figura 5.15 – Exemplo de herança de contexto

5.6. Documento de especificação de ecrãs

Com o objectivo de dar a conhecer a toda a instituição a futura aplicação, apresentar os seus ecrãs tipo, enumerar e explicar as várias funcionalidades e mostrar o *layout* dos ecrãs que servirão de base às operações frequentemente utilizadas, foi-me pedido para colaborar na elaboração de um documento que serviria como apresentação da aplicação e que seria divulgado a todos os Órgãos da ABC.

5.7. Instalação do Protótipo do processo jurídico: Elaborar Parecer

Para corresponder à fase da metodologia descrita na Secção 3.5 como Desenho, a equipa de análise desenhou os ecrãs necessários para o processo de "Elaborar Parecer" do Gabinete Jurídico e eu desenvolvi a navegação entre os vários ecrãs, apenas para o protótipo, sem ligação com a base de dados nem com o *workflow*. Como este protótipo faz parte dos *deliverables* desta fase da metodologia, foi feita uma apresentação ao cliente do mesmo, pela equipa de análise.

O processo de "Elaborar Parecer" foi assim o primeiro processo a ser apresentado ao Gabinete Jurídico e foi o primeiro contacto que os trabalhadores deste gabinete tiveram com a aplicação SiABC. Como o processo de reengenharia será um projecto longo, com várias etapas e que envolverá vários negócios dentro da ABC, esta é a fase ideal para a equipa da Novabase receber todo o *feedback* possível da parte do cliente, pois é nesta fase que se estão a fazer as definições base e a tomar decisões que implicarão toda a reengenharia. Por estas razões, decidiu-se disponibilizar uma versão deste protótipo aos vários trabalhadores do Gabinete Jurídico, nas suas próprias máquinas, para possibilitar aos utilizadores irem experimentando e testando a aplicação, com os vários ecrãs do processo.

Como o cliente é uma empresa de grande dimensão e cujo negócio implica grandes questões de segurança e de confidencialidade, toda a rede de computadores é controlada por uma empresa externa e a maioria das máquinas dos utilizadores não têm permissões de instalação de *software*.

Sendo assim, o processo de disponibilização do protótipo foi mais longo do que o que se tinha pensado inicialmente, visto que me deparei com problemas que não tinham sido antevistos. No Anexo 2, irei descrever esses problemas e soluções encontradas, retiradas de um documento entregue ao cliente, grande parte realizado por mim e nas próximas secções segue uma breve explicação dos mesmos.

5.7.1. Disponibilizar Novas Versões

Para que os utilizadores possam ter sempre acesso à versão mais recente da aplicação SiABC, foram estudadas e testadas duas possibilidades. De seguida passo a descrever cada uma delas, referindo as suas vantagens e as questões a resolver para que as próximas instalações de protótipos sejam feitas com sucesso.

1. ClickOnce

O *ClickOnce* é uma tecnologia Microsoft que nos permite "publicar" (fazer *Publish*) aplicações *Windows* num servidor *web* ou numa directoria de rede partilhada para simplificar a instalação [20]. Só está disponível a partir da versão 2.0 da *Framework.Net*.

Vantagens do *ClickOnce*:

- Permite aos utilizadores instalar e correr a aplicação a partir de uma página *html*. Neste projecto, bastaria colocar numa directoria de rede partilhada e todos os que tivessem acesso a essa directoria facilmente tinham acesso à aplicação;

- Verifica automaticamente se há novas versões, no momento em que a aplicação é executada;

- Podemos definir se está acessível *offline* (criando um *shortcut* no *All Programs*) e *online* ou só *online* (isto é, clicando no *Run* da figura 5.16).

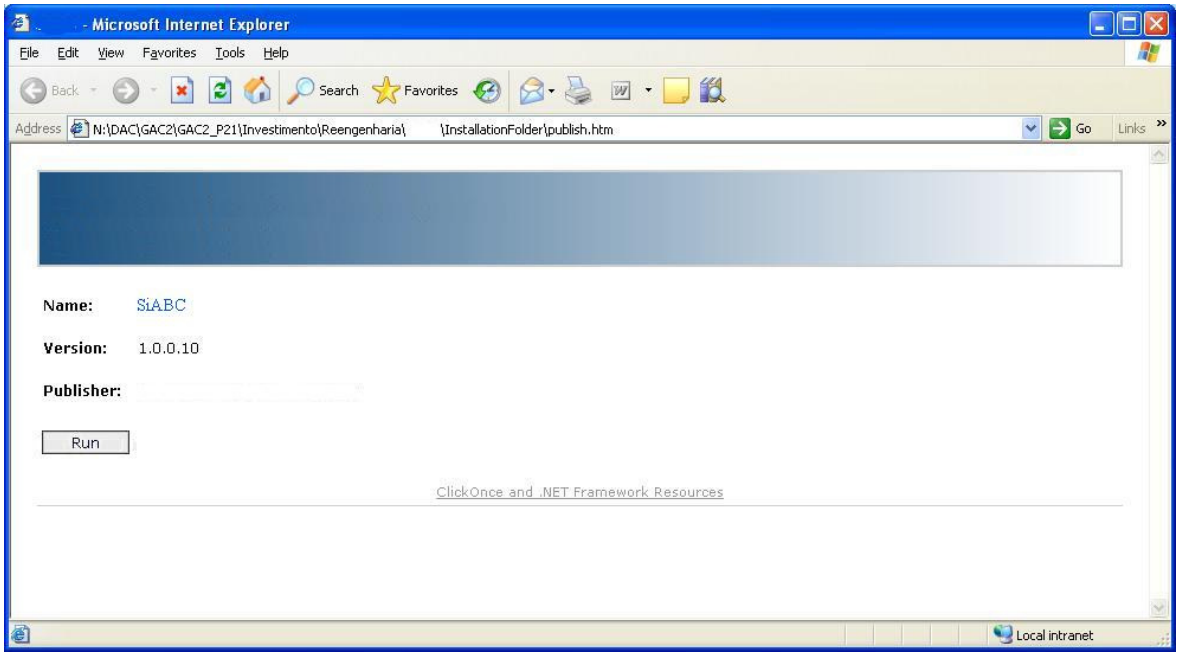

Figura 5.16 - Página *html* com a opção de correr a aplicação *online*

Questões a ter em conta:

- Para que a instalação seja feita com o *ClickOnce* é necessário que quem faz o *Publish* no *Visual Studio* tenha acesso (de escrita) a uma directoria e que todos os utilizadores da aplicação tenham acesso de leitura à mesma directoria;

- É necessário que quem faz o *Publish* tenha acesso ao *Visual Source Safe*;

- O próprio *Publish* cria um *shortcut* nas máquinas dos utilizadores e é necessário garantir que é possível criar esse *shortcut*, visto que as permissões das máquinas dos utilizadores não permitem instalar programas.

2. Directoria de Rede

Sem ser o *ClickOnce*, há a hipótese de se correr directamente o ficheiro executável (*.exe*) da aplicação.

Foi comunicado ao cliente que esta opção deverá ser considerada apenas em casos pontuais de testes. Isto porque algumas funcionalidades não têm o comportamento esperado e há o problema de a aplicação ser pesada e aumentar o tráfego de rede e a lentidão da própria aplicação. Além disso, torna mais complicada a actualização de versões: de cada vez que se quer colocar uma nova versão tem que se copiar todo o conteúdo de uma pasta e colocá-la na directoria específica.

Questões a ter em conta:

- Para correr a aplicação directamente da pasta onde se encontra o seu ficheiro *.exe*, esta tem de ter as "*Permission Set = FullTrust*". Para isso, é necessário configurar a *Framework .Net* 2.0 para dar essas permissões ao URL onde está a aplicação, em cada computador. Existe um *script* que faz isso e já foi testado na máquina de um utilizador.

- Outra hipótese será cada utilizador copiar a pasta onde está o ficheiro executável para o seu computador e corre a partir daí. Esta opção já foi utilizada para os testes do protótipo do "Elaborar Parecer".

5.7.2. Oracle Home

Na figura 5.17, pode ver-se a arquitectura do *ADO.Net*, que faz a ligação à Base de Dados

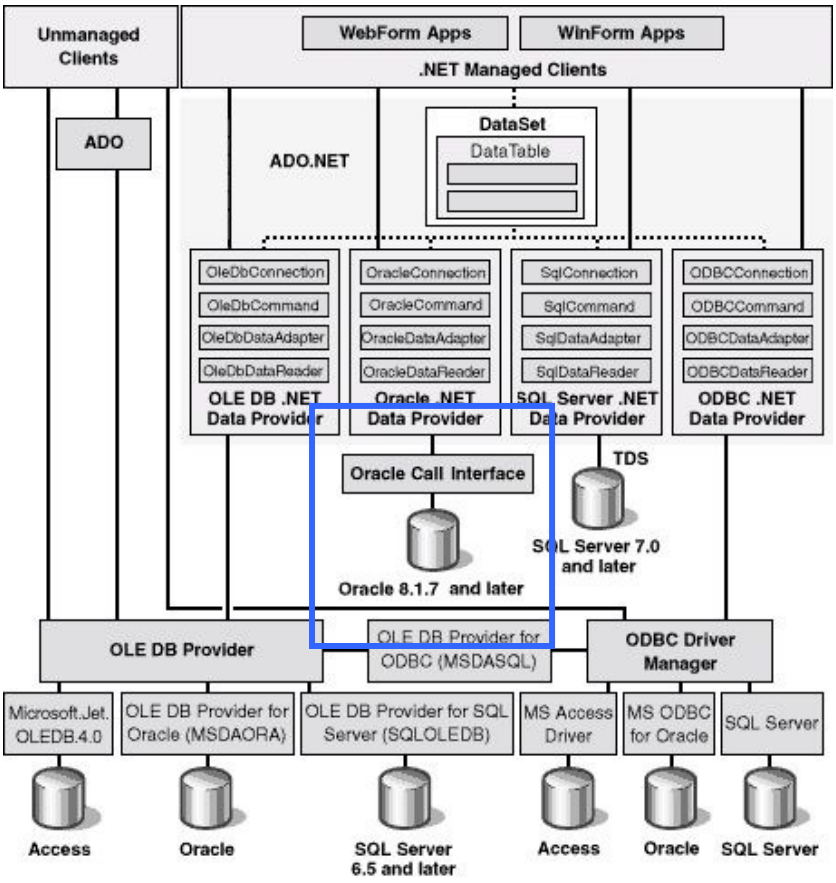

Figura 5.17 - Arquitectura do *ADO.Net*

Para desenvolver aplicações com a *Framework.Net* 2.0, podem ser usados 4 *Data Providers*, como se pode verificar na figura. Como o desenvolvimento deste projecto assenta sobre uma Base de Dados *Oracle*, o fornecedor mais optimizado é o *Oracle .NET Data Provider* que necessita, obrigatoriamente, de uma versão *Oracle* maior que 8.1.7. Esta versão tinha de estar definida no *Oracle Home* das máquinas dos utilizadores, o que não aconteceu, pois há utilizadores que precisam de trabalhar com aplicações que exigem que a versão da *Oracle Home* seja a 6.i.

5.7.3. Acesso Base de Dados

Para disponibilizar o protótipo todos os utilizadores tinham de ter acesso à base de dados de desenvolvimento, com uma ligação específica e com o *user* de testes. Como os utilizadores do JUR não utilizam nenhuma das aplicações já existentes na ABC, este acesso ainda não estava configurado.

Capítulo 6

6. Conclusões

Este projecto, no âmbito do Mestrado em Engenharia Informática, deu-me a oportunidade de trabalhar com uma grande empresa de Consultoria em Tecnologias de Informação, que me proporcionou a criação de bases sólidas para a minha evolução profissional.

O projecto consistiu na construção da plataforma base para a reengenharia das aplicações já existentes no cliente e para a inclusão de novas funcionalidades e novas áreas de negócio.

O cliente onde desenvolvi este projecto tem a particularidade de já ter uma relação de longa data e de confiança com a Novabase, por projectos anteriormente realizados. Desta forma, tinha as características necessárias para que os processos fossem bem definidos, bem planeados e bem implementados, tentando fazer deste um projecto de referência para outros dentro da área de *Advanced Costum Development*. Com estas condições, considero que foi uma mais-valia poder trabalhar e participar em todas as fases do ciclo de vida do desenvolvimento, num ambiente em que todos se empenharam bastante para seguir "à risca" uma metodologia, implementando as "melhores práticas" no que diz respeito à construção de *software*.

Outro aspecto positivo foi trabalhar com as tecnologias *state of the art* de desenvolvimento de sistemas de informação para *desktop*, tais como a *Framework.Net* da Microsoft e a linguagem C#. E também com uma ferramenta que preenche a maioria dos requisitos de uma ferramenta CASE, o *Enterprise Architect*.

Apesar de todos os aspectos positivos que advêm de um cliente com estas características, há certos pontos que podem ter trazido consequências menos desejáveis. A tomada de decisões acaba por ser um processo por vezes mais demorado, pois envolve várias pessoas e, frequentemente, várias entidades. Também o facto de terem surgido situações com um grau de prioridade mais elevado, contribuiu para que se tivesse de reformular o plano inicial e, consequentemente, as minhas tarefas.

O facto da equipa de desenvolvimento ter sofrido algumas alterações, proporcionou-me oportunidades de participar, mais activamente do que seria de esperar para o meu nível de carreira, nas decisões sobre a estrutura e a arquitectura da solução. Este facto também criou situações em que tive de resolver problemas juntamente com pessoas da equipa do cliente, e que foram resolvidos com sucesso. Foi assim um projecto que me proporcionou desafios na resoluçaõ técnica de problemas e concepção de novas abordagens e soluções.

Bibliografia

[1] Paulo Azevedo *et al*. *Ecrãs Tipo*. Novabase, 2007.

[2] Novabase. http://www.novabase.pt/showCategory.asp?idCat=EstruturaEmpresas, 17.09.2007.

[3] Novabase. *Academia Advanced Costum Development*, 2006.

[4] Nuno Estêvão. *Metodologia Orientada a Objectos*. Novabase, 2007.

[5] Software Engineering Institute. *What is a CASE Environment?*, 2004.

[6] Consultado em: http://www.powerlogic.com.br/powerportal/ecp/noticia.do?evento=portlet&pAc=not&idConteudo=3203&acao=proc&pIdPlc a 26.05.2007.

[7] *Enterprise Architect 6.5 User Guide*. Sparx Systems, 2007.

[8] OMG. *Introduction to OMG's Unified Modeling Language™*. Object Management Group Inc., 2005.

[9] Craig Larman. *Applying UML and Patterns*. Prentice Hall, 2004.

[10] Consultado em: http://en.wikipedia.org/wiki/Multiple_document_interface a 24.07.2007.

[11] Microsoft. *Microsoft Visual SourceSafe: Product Overview*. Visual Studio Developer Center, 2007.

[12] C. Mathew Mackenzie *et al*. *Reference Model for Service Oriented Architecture 1.0.* Committee Specification, 2006.

[13] Christopher Koch. *A New Blueprint For The Enterprise*. CIO Magazine, 2005.

[14] Paulo Azevedo *et al*. *Desenho de Arquitectura Orientada a Serviços (SOAD): Descrição e Regras de especificação*. Novabase, 2006.

[15] Trygve Reenskaug. *Thing-Model.View-Editor: an Example from a planningsystem*. Xerox PARC technical note, 1979.

[16] Steve Burbeck. *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*. ParcPlace Systems, 1992.

[17] Microsoft. *Model-View-Controller*. Microsoft patterns and practices Developer Center, 2007

[18] Consultado em: http://en.wikipedia.org/wiki/Model-view-controller a 3.06.2007.

[19] Consultado em: http://www.codersource.net/dataset_dot_net.html a 16.06.2007.

[20] Microsoft. *Deployment in Visual Studio: ClickOnce Deployment*. Visual Studio Developer Center, 2007.

ANEXO 1

Planeamento Inicial

Outubro a Novembro de 2006

- Formação
- Implementação da Infra-Estrutura base

Dezembro 2006 a Março 2007:

Construção dos vários processos da aplicação

- Serviços JUR

  - Tratar Solicitações ao JUR (14 - 22 Dezembro 2006)

  - Controlar Serviços do JUR (15 Dezembro 2006 – 29 Janeiro 2007)

- Documentos JUR

  - Impressão do processo de utente (29 Janeiro – 6 Fevereiro)

  - Controlar ordenação do processo de utente (6 – 20 Fevereiro)

  - Manipulação Documentos Word (20 Fevereiro – 6 Março)

- Cobrar Dívida

  - Analisar Situação para Cobrar Dívida (16 – 20 Março)

  - Iniciar forma cobrança dívida (20 Março – 24 Abril)

  - Interromper forma de cobrar Dívida (24 – 25 Abril)

- Mover Acção Judicial

  - Preparar Acção Judicial (12 – 24 Abril)

  - Elaborar Petição Inicial (24 – 26 Abril)

- Gerir Acção Judicial

  - Analisar Situação de Acção Judicial (26 – 30 Abril)

  - Preparar Diligências (30 Abril – 4 Maio)

  - Executar Diligências (4 – 22 Maio)

- Elaborar Parecer

  - Preparar Parecer (29 Janeiro – 2 Fevereiro)

  - Criar Parecer (2 – 7 Fevereiro)

- Elaborar Projecto de Diploma

  - Preparar Projecto de Diploma (6 – 12 Março)

  - Criar Projecto de Diploma (12 – 15 Março)

- Elaborar Comunicação da Direcção

  - Preparar Comunicação da Direcção (7 – 13 Fevereiro)

  - Criar Comunicação da Direcção (13 – 16 Fevereiro)

- Elaborar Ofício

  - Preparar Ofício (20 – 26 Março)

  - Criar Ofício (26 – 29 Março)

- Elaborar Certidão

  - Preparar Certidão (16 – 22 Fevereiro)

  - Criar Certidão (22 – 27 Fevereiro)

- Elaborar Memorando, Nota ou Informação

  - Preparar Memorando, Nota ou Informação (29 Março – 6 Abril)

  - Criar Memorando, Nota ou Informação (6 – 12 Abril)

- Conceitos Associados

  - Parametrização (4 Dezembro 2006 – 13 Março 2007)

  - Gerais (27 Novembro – 27 Março)

Abril a Maio 2007:

- Execução de Testes

Junho 2007:

- Elaboração do Relatório

ANEXO 2

Problemas Ocorridos na Instalação do Protótipo do processo jurídico:

"Elaborar Parecer"

Desde o início do projecto que foi comunicado ao cliente que existiam requisitos, em termos de *software,* para a correcta execução da aplicação SiABC. Estes requisitos variam conforme se esteja na óptica de desenvolvimento ou na óptica dos clientes.

Requisitos para desenvolvimento

Os requisitos de *software* para as máquinas de desenvolvimentos são os seguintes:

- *Microsoft Visual Studio Team Edition for Software Developers* – Para os programadores;

- *Microsoft Visual Studio Team Edition for Software Architects* – Para os analistas;

- *Framework.Net* 2.0;

- *Oracle Client* 8*i* ou superior;

- *Oracle Home*, seleccionada para a versão 8*i* ou superior;

Requisitos para cliente

Os requisitos de *software* para as máquinas cliente são as seguintes:

- *Framework.Net* 2.0;

- *Oracle Client* 8*i* ou superior;

- *Oracle Home*, seleccionada para a versão 8*i* ou superior;

Problemas Ocorridos na Instalação do Protótipo

2007-06-01: Foi solicitada a criação da directoria para colocar os ficheiros para a instalação do protótipo, e disponibilizados os respectivos ficheiros.

2007-06-02: Confirmação da criação da directoria.

2007-06-04: Tentativa de instalação por *ClickOnce*, não foi possível criar o *shortcut* nas máquinas dos utilizadores.

Não foi possível disponibilizar o protótipo do JUR aos utilizadores do Gabinete Jurídico pois não tinham acesso ao perfil de base de dados respectivo, nem a *framework .net* 2.0 instalada. Foi efectuado o pedido.

2007-06-18: As máquinas dos utilizadores não permitem alterar o *Oracle Home* e estão configuradas para o 6i. Como não se sabe qual o impacto da alteração nas outras aplicações, foi tomada a decisão de instalar o protótipo nas máquinas de formação e foi efectuado o pedido de alteração do *Oracle Home* só para estas máquinas.

2007-06-18: Não se conseguia executar a partir de uma directoria de rede devido a permissões. Solução encontrada: copiar para cada computador o executável.

2007-06-20: Recebemos a confirmação da alteração do *Oracle Home* nas máquinas de formação. Protótipo disponível para testes.

2007-06-29: Foi efectuado um teste numa máquina de um utilizador do Gabinete Jurídico, após a alteração do *Oracle Home* e alterações de permissões, funcionou a execução da aplicação pela directoria de rede. Falta dar o *ok* para efectuar o mesmo para todos os outros utilizadores.

2007-07-05: Foi efectuado o pedido para alterar a *Oracle Home* e as permissões de todos os utilizadores do Gabinete Jurídico.

Foi efectuado o teste com *ClickOnce* na máquina de um utilizador do Gabinete Jurídico com um *user* de formação e a aplicação foi instalada com sucesso. Com um *user* do Gabinete Jurídico não funcionou, porque necessita de permissões de leitura para a directoria onde foi efectuado o *Publish.*